# Alfresco JLAN Server

# Installation Guide

For Alfresco JLAN Server v6.0

Author: GK Spencer

# Table of Contents

# Alfresco JLAN Server Installation Guide

## 1 Alfresco JLAN Server Overview

The JLAN Server is a Java based file server implementing the Server Message Block (SMB) protocol, also known as the Common Internet File System (CIFS), File Transfer Protocol (FTP) and Network File Server (NFS) protocol.

SMB/CIFS is the protocol used by Windows networking to provide disk and print shares, plus other network administration and security functions.

The JLAN Server uses a virtual filesystem interface that provides a standard interface to the filesystem for the various protocols. The virtual filesystem may be mapped to a real filesystem, or other repository or media.

Much of the design philosophy behind the JLAN Server is about customization. Many of the key components of the system can be replaced via the main server configuration class. The key components that may be replaced/customized are:-

- Virtual filesystem driver classes
- Authentication classes
- Server configuration classes
- Virtual filesystem mapping class
- Access control manager and access control rules
- Quota manager

The JLAN Server kit contains a virtual filesystem driver class that maps to the local filesystem using the *java.io.File* class and a database filesystem that stores the filesystem structure in a database table with a custom file loader class used to load and save the file data. There are sample file loader implementations that use the local filesystem and database BLOB fields.

The demonstration server applications – *org.alfresco.jlan.app.JLANServer* and *org.alfresco.jlan.app.JLANServerService* – use an XML based server configuration implementation.

The default virtual filesystem mapping class provides access to the filesystems defined in the server configuration plus allows access to a HOME area if the user accessing the server has a home directory defined in the server configuration.

# 2 The JLAN Server Applications

The Jar file supplied with the JLAN Server kit contains two fully functional server applications that use the SMB/CIFS, NetBIOS, FTP and NFS server components:-

- org.alfresco.jlan.app.JLANServer
  Allows the JLAN Server to be started as a console application, or as an NT service.
- org.alfresco.jlan.app.JLANServerService
  Allows the JLAN Server to be started as a console application, or NT service, or Linux/Unix daemon by using the ServiceWrapper from TanukiSoftware.

The server is configured using an XML configuration file. The application uses the DOM parser that is part of the Java runtime.

The configuration file defaults to *jlanserver.xml* in the user home directory, under Windows this will be in the *Documents And Settings*\<*username*> directory. The configuration file can also be specified on the command line.

In the demo version of the JLAN Server kit the main applications are *org.alfresco.jlan.app.demo.JLANServer* and *org.alfresco.jlan.app.demo.JLANServerService*.

## 2.1 Jar Files

There are two Jar files includes in the JLAN Server kit :-

- alfresco-jlan.jar
  Contains the core server applications but does not contain the database interface code for mySQL, Oracle or Cloudscape Derby.
- alfresco-jlan-db.jar
  Contains the core server applications plus the mySQL, Oracle and Derby database interface classes.

The database filesystem version of the Jar also requires the appropriate JDBC classes to be on the classpath.

The JLAN Server requires a JCE provider that implements MD4, MD5 and DES hashing/encryption algorithms. The kit contains the Cryptix JCE provider, the Cryptix licence is reproduced below :-

```
Cryptix General License

Copyright (c) 1995-2005 The Cryptix Foundation Limited.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

  1. Redistributions of source code must retain the copyright notice,
     this list of conditions and the following disclaimer.
  2. Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in
     the documentation and/or other materials provided with the
     distribution.

THIS SOFTWARE IS PROVIDED BY THE CRYPTIX FOUNDATION LIMITED AND
```

```
CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES,
INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
IN NO EVENT SHALL THE CRYPTIX FOUNDATION LIMITED OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR
BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE
OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN
IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

Other JCE providers may be used such as Bouncy Castle, or if you are using the IBM JDK/JRE this includes the required hashing and encryption algorithms in the standard kit.

## 2.2 org.alfresco.jlan.app.JLANServer

The JLANServer application can be run as a console application or as an NT service. The following command lines show various ways that the server application can be started:-

```
java -jar alfresco-jlan.jar
java -cp .\alfresco-jlan.jar org.alfresco.jlan.app.JLANServer jlanconfig.xml
java -jar .\alfresco-jlan.jar jlanConfig.xml
```

A sample configuration file is included in the demo kit (*jlanserver.xml*). The sample configuration file is setup to use the Win32 NetBIOS interface.

To use the NetBIOS over TCP/IP and/or native SMB interfaces the network broadcast mask must be configured before the sample configuration file can be used.

The *runsrv.bat* batch file may also be used to start the server under Windows.

## 2.3 org.alfresco.jlan.app.JLANServerService

The JLANServerService application uses the ServiceWrapper from TanukiSoftware (http://wrapper.tanukisoftware.org/) to provide portability and resilience.

The ServiceWrapper is available for a wide range of platforms, including Windows, Linux, Mac OS X, Irix, HP-UX, Aix, FreeBSD and Solaris.

The JLAN Server kit contains the binaries for Windows, Linux, Solaris and Mac OS X support in the *service*\ sub-directory. A pre-configured ServerWrapper configuration file is included – jlansrv.conf. The main JLAN Server configuration file is expected to be in the user home directory, the jlansrv.conf file only provides the ServiceWrapper configuration of the JVM, application class, logging, JVM monitoring and NT service parameters.

To start the JLANServerService under Windows use the JLANServer.exe in the *wrapper**windows* sub-directory. To start as a console application use the following command line:-

```
jlanserver -c jlansrv.conf
```

The ServiceWrapper can also be used to run the JLAN Server as an NT service or daemon process. To install and start the JLAN Server as an NT service use the following commands:-

```
jlanserver -i jlansrv.conf
jlanserver -t jlansrv.conf
```

The *wrapper.ntservice.account* and *wrapper.ntservice.password* parameters in the jlansrv.conf will need to be modified before installing the JLAN Server as an NT service.

To start the JLANServerService under Linux, Solaris or FreeBSD use the jlanserver application in the appropriate *wrapper*\ sub-directory.

A script is provided in the *service**linux*\ sub-directory that can be used to start/stop/restart the JLAN Server as a daemon process under Linux.

For more information on configuring the ServiceWrapper and to download support for other operating systems visit the TanukiSoftware web site at http://wrapper.tanukisoftware.org/.

## 2.4 JLAN Server XML Configuration File

The JLAN Server is configured using a simple XML file. A DTD is available in the kit to validate the configuration (*jlanserver.dtd*).

The configuration is contained within the <*jlanserver*> section of the configuration file. The server is configured via the <*servers*>, <*global*>, <*SMB*>, <*FTP*>, <*NFS*> <*shares*>, <*security*>, <*shareMapper*>, <*DriveMappings*>, <*cluster*> and <*debug*> sub-sections.

Configuration items added in recent versions of the JLAN Server are shown in bold type.

### 2.4.1 Server Configuration

The <*servers*> section defines the various protocol servers that are to be enabled.

| **Server Configuration** | |
|---|---|
| <SMB/> or <CIFS/> | Enable the SMB server |
| <FTP/> | Enable the FTP server |
| <NFS/> | Enable the NFS server (mount server and main NFS server) |

### 2.4.2 Global Configuration

The <*global*> section defines the settings that are global to the server configuration and various protocols.

| **Global Configuration** | |
|---|---|
| <timezone name="..."/><br><timezone offset="..."/> | Specifies the server timezone using either the name, such as 'GMT' or 'PST' with the *name* attribute, or by specifying the offset from UTC in minutes with the *offset* attribute.<br><br>The *offset* may be specified as a positive or negative value. |

| Global Configuration | |
|---|---|
| | <timezone name="PST"/><br><timezone offset="-480"/> |

## 2.4.3 SMB Server Configuration

The <*SMB*> section defines the SMB/CIFS server configuration details and network configuration. The main host configuration is contained within the <*host*> sub-section, with various debug settings being specified by the <*sessionDebug*>, <*netbiosDebug*> and <*announceDebug*> items.

| Host Configuration | |
|---|---|
| <host name="..." domain="..."/> | Specifies the server name and domain/workgroup that the server is part of.<br><br><host name="JLANSRV" domain="STARLASOFT"/> |
| <alias names="name1,name2,.."/> | Specifies alias names that the server will respond to. |
| <broadcast>n.n.n.n</broadcast> | Network or subnet broadcast mask as a dotted TCP/IP address. In some cases the value of '255.255.255.255' may work.<br><br><broadcast>90.1.255.255</broadcast> |
| <smbdialects>..,..</smbdialects> | Enables the SMB dialects that the server will negotiate with a client. The available dialects are *Core*, *LanMan* and *NT*.<br><br><smbdialects>Core,LanMan,NT</smbdialects> |
| <comment>...</comment> | Server comment sent out as part of the host announcement and also returned by various server/workstation information requests.<br><br><comment>JLAN SMB Server</comment> |
| <bindto>n.n.n.n</bindto><br><bindto adapter="..."/> | Specifies the network adapter to bind to if the host has multiple network adapters. If not specified the SMB server will bind to all available adapters.<br>Alternatively, the adapter name may be specified using the adapter attribute. The adapter name is the name returned by the NetworkInterface class, such as 'eth0' or 'en0'.<br><bindto>90.1.0.0</bindto><br><bindto adapter="eth0"/> |
| <authenticator type="..."><br>..<br></authenticator> | Configures the CIFS server authentication mechanism.<br>The *type* attribute specifies that one of the supplied |

| **Host Configuration** | |
|---|---|
| <authenticator><br>  <class>...</class><br>  ..<br></authenticator> | authenticators be used. The available values are 'local' for the simple authenticator that supports NTLM1 logons, 'passthru' for the passthru authenticator or 'enterprise' for the authenticator that supports NTLM v1, NTLM v2 and Kerberos logons.<br>A custom authenticator can be specified by using the <class> configuration tag. Other configuration tags may be specified as required. |
| <netBIOSSMB/><br><br><netBIOSSMB bindto="n.n.n.n"/><br><br><netBIOSSMB adapter="..."/><br><br><netBIOSSMB platforms="..."/> | Enables the NetBIOS over TCP/IP protocol on port 139.<br><br>To run the JLAN SMB Server under Windows you must disable NetBIOS over TCP/IP via the Network Control Panel.<br><br>The *bindto* attribute may be specified to bind the NetBIOS name server to a particular address when the system has multiple network adapters.<br>Alternatively, the adapter name may be specified using the adapter attribute. The adapter name is the name returned by the NetworkInterface class, such as 'eth0' or 'en0'.<br>The *platforms* attribute may be specified to control which platforms the NetBIOS SMB component will be enabled on. The platforms value is a comma delimeted list of platform names where the valid names are *linux*, *macosx*, *windows, solaris* and *aix*.<br><br>NetBIOS over TCP/IP may be enabled at the same time as the native SMB over TCP/IP protocol. |
| <tcpipSMB/><br><br><tcpipSMB platforms="..."/><br><br><tcpipSMB ipv6="enabled"/> | Enables the native SMB over TCP/IP protocol on port 445.<br><br>The *platforms* attribute may be specified to control which platforms the NetBIOS SMB component will be enabled on. The platforms value is a comma delimeted list of platform names where the valid names are *linux*, *macosx*, *windows, solaris* and *aix*.<br><br>The *ipv6* attribute enables native SMB support under Ipv6 sockets. The JLAN Server will bind to IPv4 and IPv6 sockets.<br>To run the native SMB over TCP/IP protocol under Windows you need to disable Windows from using the port via the following registry key:- |

| **Host Configuration** | |
|---|---|
| | [HKLM\SYSTEM\CurrentControlSet\Services\NetBT\Parameters]<br>"SMBDeviceEnabled"=dword:00000000<br><br>Native SMB over TCP/IP may be enabled at the same time as NetBIOS over TCP/IP.<br><br>The kit contains a registry file ( *port445.reg*) which can be used to disable the Windows file server on port 445. |
| <Win32NetBIOS name="..." accept="..." lana="n" api="..."/> | Enables the Win32 NetBIOS native interface protocol that uses the Win32 Netbios() API call to provide the naming, session and datagram support.<br><br>The *name* attribute specifies the server name to accept connections on. If not specified the main SMB server name is used.<br><br>The *accept* attribute can be used to restrict the clients that are allowed to connect to the server. This is useful if the JLAN Server should only be accessible from the local host.<br><br>The *lana* attribute can be used to specify which NetBIOS LAN adapter the Win32 NetBIOS interface will use. If not specified the first available LANA will be used.<br><br>The *api* attribute is used to specify the native code interface to be used. The valid values are *netbios* for the original Win32 Netbios() API based code or *winsock* for the new Winsock Netbios based code.<br>The default is to use the Winsock NetBIOS code. |
| <WINS><br><primary>...</primary><br><secondary>...</secondary><br></WINS> | Enables the NetBIOS name server to use the specified WINS server(s) when registering the local NetBIOS names.<br><br>The secondary WINS server address is optional. |
| <hostAnnounce interval='n'/> | Enables host announcement so that the JLAN SMB server appears under Network Neighborhood. Host announcements will only be sent out if the NetBIOS over TCP/IP protocol is enabled via the <netBIOSSMB/> configuration item.<br><br>The *interval* attribute specifies the announcement |

| Host Configuration | |
|---|---|
| | interval in minutes.<br><br><hostAnnounce interval="5"/> |
| <HostAnnouncerPort><br>n<br></HostAnnouncerPort> | Specifies the datagram port to be used by the host announcer when sending announcement datagrams.<br><br>If not specified the default port of 138 will be used. |
| <Win32Announce interval="n"/> | Enables host announcement via the Win32 Netbios API so that the JLAN SMB server appears under Network Neighborhood.<br><br>The *interval* attribute specifies the announcement interval in minutes. |
| <sessionTimeout><br>n<br></sessionTimeout> | Specifies the CIFS session timeout value in seconds. The default session timeout is 15 minutes.<br><br>If no I/O occurs on the session within this time then the session will be closed by the server. Windows clients send keep-alive requests, usually within 15 minutes. |
| <disableNIO/> | Disables the new NIO based CIFS server code and reverts to using the older socket and JNI based code. |

The SMB server has many debug settings which are controlled by the following configuration items:-

| SMB Server Debug Configuration | |
|---|---|
| <sessionDebug flags="..."/> | Enables various SMB session level debug output. See the table below for the list of available debug levels.<br><br><sessionDebug flags="NetBIOS,File,IPC,Error"/> |
| <netbiosDebug/> | Enables NetBIOS name server debug output. |
| <announceDebug/> | Enables host announcer debug output. |

The following table lists the available session debug levels:-

| **Session Debug Levels** | |
|---|---|
| NETBIOS | Name session debugging |
| STATE | Session state changes |
| RXDATA | Received session data |
| TXDATA | Transmitted session data |
| ERROR | Request errors |
| NEGOTIATE | SMB dialect negotiation |
| TREE | Share connection/disconnection |
| SEARCH | File/directory searches |
| INFO | Information requests |
| FILE | File access |
| FILEIO | File read/write |
| TRANSACT | SMB transactions |
| ECHO | Client session keep-alive |
| IPC | IPC$ named pipe requests |
| PKTTYPE | Output received packet type |
| DCERPC | DCE/RPC handling |
| NOTIFY | Change notification processing |
| STREAMS | NTFS streams |
| SOCKET | Low level connections |
| LOCK | File byte range locks/unlocks |
| STATECACHE | File state caching |
| TIMING | Request/response timing |
| PKTPOOL | Memory pool allocations/deallocations |
| PKTSTATS | Dump memory pool statistics during server shutdown |
| THREADPOOL | Thread pool |
| BENCHMARK | Benchmarking |
| **STATECACHE** | File state cache |
| **OPLOCK** | Oplocks |

A sample SMB configuration section is shown below:-

```
<SMB>
  <host name="JLANSRV" domain="STARLASOFT">
    <broadcast>192.168.1.255</broadcast>
    <smbdialects>Core,LanMan,NT</smbdialects>
    <comment>JLAN SMB Server</comment>
    <bindto>192.168.1.2</bindto>
```

```
    <netBIOSSMB/>
    <hostAnnounce interval="5"/>
  </host>
  <sessionDebug flags="Negotiate,Tree"/>
  <netbiosDebug/>
  <announceDebug/>
</SMB>
```

## 2.4.4 Cluster Configuration

The <cluster> section defines the global cluster configuration.

The JLAN Server currently has clustered file state cache and debug interface classes which are based on the Hazelcast clustering code. The <cluster> configuration section allows a single Hazelcast instance to be used by multiple clustered filesystems and/or have debug output piped to a central server.

| **Cluster Configuration** | |
|---|---|
| <configFile><br>…<br></configFile> | Specifies the path of the Hazelcast XML configuration file that is used to create the shared Hazelcast instance. |

## 2.4.5 FTP Server Configuration

The <*FTP*> section defines the FTP server configuration.

| **FTP Configuration** | |
|---|---|
| <authenticator><br><class>...</class><br>..<br></authenticator> | The authenticator configuration sub-section is used to enable a custom FTP authentication class that implements the *org.alfresco.jlan.ftp.FTPAuthenticator* interface.<br>The *class* must be specified, other configuration parameters may be specified as required. |
| <bindto>n.n.n.n</bindto><br><bindto adapter="..."/> | Specifies which network adapter to bind to if the host has multiple network adapters. If not specified the FTP server will bind to all available adapters.<br>Alternatively, the adapter name may be specified using the adapter attribute. The adapter name is the name returned by the NetworkInterface class, such as 'eth0' or 'en0'.<br><bindto>192.168.1.2</bindto><br><bindto adapter="eth0"/> |
| <port>n</port> | Specifies the port that the FTP server listens for incoming connections on. The default port is 21. |
| <rootDirectory><br>..<br></rootDirectory> | Specifies the path to be used for the root directory when a client connects to the FTP server.<br>The root directory path must be specified using the FTP |

| FTP Configuration | |
|---|---|
| | path format, using forward slashes in the path '/'.<br><br>The root directory path may specify just the virtual filesystem to be used as the root, ie. /name, or may contain one or more sub-directories on the particular virtual filesystem, ie. /name/dir1/dir2. |
| <dataPorts rangeFrom=”n” rangeTo=”n”/> | Allows a port range to be specified to control the data ports that the FTP server may open. |
| <allowAnonymous/> | Allow anonymous access to the FTP server. If not enabled only defined users will be able to access the FTP server. |
| <debug flags=”...”/> | Enables various FTP server/session debug output. |
| <siteInterface><br><class>..</class><br>..<br></siteInterface> | The site interface configuration sub-section is used to enable site specific extensions to the FTP server.<br>The class must be specified, and must implement the *org.alfresco.jlan.ftp.FTPSiteInterface* interface. Other configuration parameters may be specified as required. |
| **<keyStore>**<br>**...**<br>**</keyStore>** | Path to the keys store file when FTPS is enabled. |
| **<trustStore>**<br>**...**<br>**</trustStore>** | Path to the trust store file when FTPS is enabled. |
| **<storePassphrase>**<br>**...**<br>**</storePassphrase>** | Store passphrase |
| **<requireSecureSession/>** | Only allow sessions that use an FTPS logon. |
| **<sslEngineDebug/>** | Enable FTPS SSL engine debug output. |

The following table lists the available FTP debug levels:-

| FTP Debug Levels | |
|---|---|
| STATE | Session state changes |
| RXDATA | Received session data |
| TXDATA | Transmitted session data |
| SEARCH | File/directory searches |
| INFO | Information requests |
| FILE | File access |
| FILEIO | File read/write |
| ERROR | Request errors |
| PKTTYPE | Request types |
| DATAPORT | Data session |
| DIRECTORY | Directory related commands |

| FTP Debug Levels | |
|---|---|
| TIMING | Request/response timing |
| SSL | FTPS/SSL |

A sample FTP server configuration section is shown below:-

```
<FTP>
  <bindto>192.168.1.2</bindto>
  <allowAnonymous/>
  <debug flags="File,FileIO,Search,Error"/>
</FTP>
```

## 2.4.6 NFS Server Configuration

The <*NFS*> section defines the NFS server configuration.

| **NFS Configuration** | |
|---|---|
| <enablePortMapper/> | Enable the port mapper service. |
| <MountServerPort><br>n<br></MountServerPort> | Specifies the port to be used by the mount server for UDP and TCP requests.<br><br>If not specified the next available port will be allocated to the mount server. |
| <PortMapperPort><br>n<br></PortMapperPort> | Specifies the port to be used by the port mapper for UDP and TCP requests.<br><br>If not specified the default port of 111 will be used. |
| <NFSServerPort><br>n<br></NFSServerPort> | Specifies the port to be used by the NFS server for UDP and TCP requests.<br><br>If not specified the default port of 2049 will be used. |
| <PacketPool><br>n<br></PacketPool> | Specifies the number of RPC request packet buffers to allocate for request processing. The value must be at least equal to the thread pool size plus one.<br><br>The packet pool should be two to three times the thread pool size for optimum performance. |
| <ThreadPool><br>n<br></ThreadPool> | Number of worker threads to allocate to the thread pool that processes RPC requests from NFS clients. The thread pool is now shared between the TCP and UDP connections.<br>Defaults to 16 worker threads. |
| <debug flags="..."/> | Enable various NFS server debug output. |
| <mountServerDebug/> | Enables mount server debug output. |
| <portMapperDebug/> | Enables port mapper server debug output. |
| <rpcAuthenticator><br><class>...</class><br></rpcAuthenticator> | Specifies the class to be used to provide RPC authentication to the mount and NFS servers.<br><br>The class must implement the *org.alfresco.jlan.oncrpc.RpcAuthenticator* interface.<br><br>A default RPC authenticator that allows any client to access the RPC servers is used if no authenticator is specified. |
| **<disablePortMapperRegistration/>** | Do not register the NFS and mount servers with a port mapper service. |
| **<FileCache>n[:m]</FileCache>** | File cache timer value(s) in seconds. Specifies the amount of time to keep a file open after an I/O or close request is |

| NFS Configuration | |
|---|---|
| | received.<br>If only a single value is specified it will be used for file I/O and file close timers.<br>Two values may be specified, seperated by a ':'. The first value is the I/O timer and the second value is the file close timer, in seconds. |
| **<fileCacheDebug/>** | Enable file cache debug output. |

The following table lists the available NFS debug levels:-

| NFS Debug Levels | |
|---|---|
| RXDATA | Received session data |
| TXDATA | Transmitted session data |
| SEARCH | File/directory searches |
| INFO | Information requests |
| FILE | File access |
| FILEIO | File read/write |
| ERROR | Request errors |
| DIRECTORY | Directory related commands |

A sample NFS server configuration section is shown below:-

```
<NFS>
  <enablePortMapper/>
  <debug flags="File,Fileio,Search,Mount"/>
</NFS>
```

## 2.4.7 Shares Configuration

The <shares> section defines the available virtual filesystems. Each virtual filesystem is associated with a driver class that provides the interface between the virtual filesystem and the core protocol servers.

The JLAN Server Jar file contains two virtual filesystem drivers:-

- JavaFileDiskDriver
  Maps the virtual filesystem to the local filesystem using the java.io.File class.
- DBDiskDriver
  Uses a database to hold the virtual filesystem structure. The database interface used is configurable to allow different database types to be used. The file data is accessed via a file loader class with different implementations allowing the file data to be stored on the local filesystem, in database BLOB fields or in a repository.

| **Shares Configuration** | |
|---|---|
| <diskshare name="..." comment="..."/> | Defines a virtual filesystem. The *name* attribute specifies the share name that will be used by a client to map or mount the virtual filesystem.<br>The optional *comment* is returned by various information requests.<br><br><diskshare name="JLAN" comment="Test area"/> |
| <driver> | Specifies the start of the disk share driver class definition block. |
| <accessControl><br><accessControl default="..."> | Specifies the access control rules block.<br><br>The *default* attribute specifies the default access for clients that do not match any of the access control rules. The default value may be *Read* for read-only access, *Write* for read/write access or *None* for no access.<br><br>An empty access control block may be specified with a default value or *Read* or *Write*. |
| <disableChangeNotification/> | Disables the processing of change notifications for this virtual filesystem.<br><br>Windows clients register for change notification to watch file/directory changes made by other clients. |
| <size totalSize="n" freeSize="n"/><br><br><size totalSize="n" freeSize="n" blockSize="n" blocksPerUnit="n"/> | Specifies the virtual disk size, free space and optionally the block and allocation unit sizes.<br><br>The disk size and free space may be specified as 'n' bytes, 'nK' for kilobytes, 'nM' for megabytes or 'nT' for terabytes. |

| Shares Configuration | |
|---|---|
| | The blockSize defaults to 512 bytes and the blocksPerUnit defaults to 64 to indicate a 32Kb allocation unit. It is recommended that these values are not altered.<br><br>The virtual filesystem driver class may implement the *DiskSizeInterface* to provide dynamic disk size information. |
| <volume label="..." serial="n" created="d"/> | Specifies the virtual disk volume label, and optionally the serial number and creation date/time.<br><br>The creation date/time string should be in the 'dd-MMM-yyyy' or 'dd-MMM-yyyy hh:mm:ss' format.<br><br>The virtual filesystem driver class may implement the *DiskVolumeInterface* to provide the volume information programatically. |

The <driver> sub-section contains the virtual filesystem driver class details and driver specific configuration values. The <driver> section must contain a <class>...</class> item to specify the driver class, for example:-

```
<class>org.alfresco.jlan.smb.server.disk.JavaFileDiskDriver</class>
```

The <accessControl> sub-section contains the access control rules that are used to allow read or read/write access to the share, or to disallow access to the share.

The <accessControl> block may be empty if a default access of Read or Write is specified, for example :-

```
<accessControl default="Read"/>
```

The following table details the access control rules that are available via the default access control manager. The *access* attribute may have the value *Read* for read-only access, *Write* for read/write access or *None* to disallow access.

| Access Control Rule | Description |
|---|---|
| <user name="..." access="..."/> | Set the access for the specified user.<br><br>The rule applies to SMB/CIFS and FTP sessions. |
| <protocol type="..." access="..."/> | Set the access depending upon the protocol used by the client.<br><br>The *type* attribute may contain a comma delimeted list of protocol names. Valid protocol names are *SMB*, *CIFS*, *NFS* and *FTP*. |
| <address subnet="..." mask="..." access="..."/> | Set the access depending upon the clients network address.<br><br>The *subnet* attribute specifies the network |

| Access Control Rule | Description |
|---|---|
| | subnet in n.n.n.n format.<br>The *mask* attribute specifies the network mask in n.n.n.n format. |
| <address ip="..." access="..."/> | Set the access depending upon the clients network address.<br>The *ip* attribute specifies the client address in n.n.n.n format. |
| <domain name="..." access="..."/> | Set the access depending on the callers domain name.<br>This rule only applies to SMB/CIFS sessions. |

A sample access control block is shown below :-

```
<diskshare name="TESTAREA">
  <driver>
    <class>
      org.alfresco.jlan.smb.server.disk.JavaFileDiskDriver
    </class>
    <LocalPath>N:\TestArea</LocalPath>
  </driver>

  <volume label="TESTLABEL"/>
  <size totalSize="2T" freeSize="100G"/>

  <accessControl default="Read">
    <user name="gkspencer" access="Write"/>
    <user name="GK Spencer" access="Write"/>
    <address subnet="192.168.1.0" mask="255.255.255.0" access="Write"/>
    <domain name="LAPTOP" access="None"/>
  </accessControl>
</diskshare>
```

### 2.4.7.1 JavaFileDiskDriver Configuration

The *JavaFileDiskDriver* class maps a virtual filesystem to the local filesystem using the *java.io.File* class. The *JavaFileDiskDriver* class is in the *org.alfresco.jlan.smb.server.disk* package.

The <*driver*> sub-section configuration parameters are shown below:-

| **JavaFileDiskDriver Configuration** | |
|---|---|
| <LocalPath>...</LocalPath> | Specifies the local path to map the virtual filesystem to. |

A sample JavaFileDiskDriver share configuration section is shown below:-

```
<shares>
  <diskshare name="JLAN" comment="Test share">
    <class>
      org.alfresco.jlan.smb.server.disk.JavaFileDiskDriver
    </class>
    <LocalPath>R:\JLAN</LocalPath>
  </diskshare>
</shares>
```

### 2.4.7.2 DBDiskDriver Configuration

The *DBDiskDriver* class holds the virtual filesystem structure details in a database table.The *DBDiskDriver* class is in the *org.alfresco.jlan.server.filesys.db* package.

The DBDiskDriver does not contain any database access code, it relies on a seperate database interface class to provide the database persistence. The database interface does not need to be JDBC or SQL based.

The following database interface implementations are included in the JLAN Server kit :-

| Database | Interface Class |
|---|---|
| MySQL | org.alfresco.jlan.server.filesys.db.mysql.MySQLDBInterface |
| Oracle | org.alfresco.jlan.server.filesys.db.oracle.OracleDBInterface |
| Derby | org.alfresco.jlan.server.filesys.db.derby.DerbyDBInterface |

The DBDiskDriver uses a seperate interface to load and save the file data, a file loader. The file loader implementation may allow direct access to the file data or may use a thread pool of worker threads to load/save the file data from a repository.

The following file loaders are included in the JLAN Server kit :-

| Name | Description |
|---|---|
| SimpleFileLoader | Provides a simple file loader that loads/saves files to the local filesystem maintaining the same directory structure.<br>The loader class is *org.alfresco.jlan.server.filesys.loader.SimpleFileLoader*. |
| DBFileLoader | Loads/saves file data to database BLOB fields using a thread pool of worker threads to load/save the file data in background. A queue of load/save requests is maintained in a database table for crash recovery.<br>The loader class is *org.alfresco.jlan.server.filesys.db.DBFileLoader*. |

The DBDiskDriver can use the clustered file state cache to allow multiple JLAN Servers to operate as a cluster, where file locks, access mode checks and updates are implemented between the cluster members.

The following table lists the main DBDiskDriver configuration sections :-

| DBDiskDriver Configuration | |
|---|---|
| <DatabaseInterface><br>..<br></DatabaseInterface> | Specifies the database interface implementation to be used by the DBDiskDriver. |
| <FileLoader>..</FileLoader> | Specifies the file loader interface to be used to load and save the file data. |
| <disableNTFSStreams/> | Disable NTFS streams support.<br>NTFS streams support is also dependent on the file loader implementation. |
| <enableTrashCan/> | Enables the trashcan that marks files/folders as deleted |

| DBDiskDriver Configuration | |
|---|---|
| | rather than deleting them from the database. |
| <QuotaManagement/> | Enables quota management for this share.<br>Disk quotas are enforced using the disk size value set via the <size ...> configuration value. |
| <RetentionPeriod><br>...<br></RetentionPeriod> | Enables file/folder retention. The configuration value specifies the retention period as the number of days.<br><RetentionPeriod>7</RetentionPeriod> |
| **<disableOplocks/>** | Disable oplock support for this shared filesystem. |
| **<stateCache type="...">**<br>**...**<br>**</stateCache>** | Specifies the type of file state cache to be used by the filesystem. The default state cache is a standalone cache for use on a single server. A clustered state cache may be used when a number of servers are operating as a cluster.<br>The state cache types available are 'standalone' for the default standalone server cache, 'cluster' for the clustered cache or 'custom' for a custom cache implementation. |

The standalone state cache has the following configuration values :-

| Standalone State Cache Configuration | |
|---|---|
| <fileStateExpire><br>n<br></fileStateExpire> | Specifies the file state expiry interval, in seconds. This is the number of seconds a file state may be held in the state cache after the file has been closed by the last referencing session.<br>The default value is 300 seconds (5 minutes). The minimum allowed value is 15 seconds. |
| <cacheCheckInterval><br>n<br></cacheCheckInterval> | Specifies how often the file state expiry check is run, in seconds.<br>The default value is 60 seconds, the minimum allowed value is 5 seconds. |
| <Debug/> | Enable file state cache debug output. |
| <expiryDebug/> | Enable additional file state expiry debug output. |
| <initialSize><br>n<br></initialSize> | Specifies the initial allocation size of the hash table used by the file state cache.<br>The default value is 500 elements, the minimum allowed size is 100 elements. |

The clustered state cache has the following configuration values :-

| **Clustered State Cache Configuration** | |
|---|---|
| <fileStateExpire><br>n<br></fileStateExpire> | Specifies the file state expiry interval, in seconds. This is the number of seconds a file state may be held in the state cache after the file has been closed by the last referencing session.<br>The default value is 300 seconds (5 minutes). The minimum allowed value is 15 seconds. |
| <cacheCheckInterval><br>n<br></cacheCheckInterval> | Specifies how often the file state expiry check is run, in seconds.<br>The default value is 60 seconds, the minimum allowed value is 5 seconds. |
| <clusterName><br>…<br></clusterName> | Specifies the map name entry from the Hazelcast XML configuration file (specified using the top level <cluster> configuration section). |
| <clusterTopic><br>…<br></clusterTopic> | Specifies the topic name used to send messages between nodes in the cluster. |
| <nearCache disable/><br><nearCache timeout=”n”/> | Used to disable the near cache or set the near cache timeout value.<br>If the near cache is disabled clustered state cache lookups may require a network access to fetch the required file state.<br>The default near cache timeout value is 5 seconds, the minimum allowed value is 3 seconds, the maximum allowed value is 120 seconds (2 minutes). |
| <cacheDebug flags=”...”/> | Enable various clustered file state cache debug output. |

The following table lists the available clustered file state cache debug levels :-

| **Clustered File State Cache Debug Levels** | |
|---|---|
| STATECACHE | Cache get/put/find |
| EXPIRE | Cache expiry |
| NEARCACHE | Near cache get/put/hits |
| OPLOCK | Oplock grant/release |
| BYTELOCK | Byte range lock/unlock |
| FILEACCESS | File access grant/release |
| MEMBERSHIP | Cluster membership changes |
| CLEANUP | Cleanup when a node leaves the cluster |
| PERNODE | Per node updates |
| CLUSTERENTRY | Cluster entry updates |
| CLUSTERMESSAGE | Cluster messaging |
| REMOTETASK | Remote tasks |
| REMOTETIMING | Remote task timing, key lock/unlock |

| | |
|---|---|
| RENAME | Rename state |
| FILEDATAUPDATE | File data updates |
| FILESTATUS | File status changes (exist/not exist) |

### 2.4.7.2.1 DatabaseInterface Configuration

The database interface classes included in the JLAN Server kit have the following configuration parameters :-

| **DatabaseInterface Configuration** | |
|---|---|
| <DSN>...<DSN> | Specifies the datasource that holds the JLAN Server database tables.<br>The required database tables will be created if they do not exist.<br><DSN>jdbc:derby:DerbyDB;create=true</DSN> |
| <UserName>...</UserName> | Username required to access the datasource. |
| <Password>...</Password> | Password required to access the datasource. |
| <ConnectionPool><br>...<br></ConnectionPool> | Specifies the database connection pool size. The value may be specified as 'n' maximum connections or 'n:m' for initial and maximum connections.<br>The default is a minimum of 5 connections and maximum of 10 connections. |
| <Debug/> | Enable DBDiskDriver debug output. |
| <SQLDebug/> | Enable output of SQL statements. |
| <FileSystemTable><br>...<br></FileSystemTable> | Name of the database table that holds the main filesystem structure details.<br>Defaults to *JLANFileSys* if not specified. |
| <StreamsTable><br>...<br></StreamsTable> | Name of the database table that holds the NTFS streams details.<br>Defaults to *JLANStreams* if not specified. |
| <RetentionTable><br>...<br></RetentionTable> | Name of the database table that holds the file/folder retention details.<br>Defaults to *JLANRetain* if not specified. |
| <QueueTable><br>...<br></QueueTable> | Name of the database table used to hold the background load/save request details.<br>Default to *JLANQueue* if not specified.<br>When running in a cluster configuration each node must use a seperate queue table. |
| <TransactionQueueTable><br>...<br></TransactionQueueTable> | Name of the database table that holds the background load/save transaction requests.<br>Defaults to *JLANTransQueue* if not specified.<br>When running in a cluster configuration each node must use a seperate queue table. |
| <DataTable><br>...<br></DataTable> | Name of the database table that holds the file data when using the DBFileLoader.<br>Defaults to *JLANData* if not specified. |
| <JarDataTable><br>...<br></JarDataTable> | Name of the database table that holds the Jar file data containing multiple small files.<br>Defaults to *JLANJarData* if not specified. |

### 2.4.7.2.2 FileLoader Configuration

The file loader specific configuration is contained within the <*FileLoader*> sub-section.

The *SimpleFileLoader* has the following configuration parameters:-

| **SimpleFileLoader Configuration** | |
|---|---|
| <class>..</class> | Specifies the file loader class, which must be an implementation of the *org.alfresco.jlan.server.filesys.loader.FileLoader* interface.<br><br>For the *SimpleFileLoader* the class is *org.alfresco.jlan.server.filesys.loader.SimpleFileLoader*. |
| <RootPath>..</RootPath> | Specifies the path on the local filesystem to use to store the files/directories.<br><br><RootPath>P:\JDBCShare</RootPath> |
| <Debug/> | Enables SimpleFileLoader debug output. |

The DBFileLoader has the following configuration parameters:-

| **JDBCFileLoader Configuration** | |
|---|---|
| <class>..</class> | Specifies the file loader class, which must be an implementation of the *org.alfresco.jlan.smb.loader.FileLoader* interface.<br><br>For the JDBC*FileLoader* the class is *org.alfresco.jlan.smb.disk.jdbc.JDBCFileLoader*. |
| <FragmentSize><br>n<br></FragmentSize> | Specifies the maximum size of file data to be stored per blob. If the file is larger than this value it will be split up into multiple blob fields.<br><br>The value may be specified as 'nK' for kilobytes or 'nM' for megabytes.<br><br>The default size is 512K, minimum allowed is 64K. |
| <ThreadPoolSize><br>l[:s]<br></ThreadPoolSize> | Number of threads to be allocated to process the background file data load/save requests.<br><br>Separate thread pools are used for load and save request processing. If only one value is specified each threadpool allocates that many worker threads.<br><br>Different thread pool sizes may be specified for load and save by specifying the value as |

| JDBCFileLoader Configuration | |
|---|---|
| | *<load_threads>:<save_threads>.*<br><br>The default number of worker threads is 4. |
| <TempDirectory><br>..<br></TempDirectory> | Specifies the local directory to be used to cache the file data whilst the file is being accessed. |
| <MaximumFilesPerDirectory><br>n<br></MaximumFilesPerDirectory> | Specifies the number of temporary files to store per sub-directory within the temporary file cache area. This is to prevent performance problems when a directory contains several thousand files. |
| <MemoryQueueSize><br>n<br></MemoryQueueSize> | Specifies the number of load/store file requests to hold in memory from the persistent file load/store queue database table.<br><br>The default size is 200 requests. |
| <QueueLowWaterMark><br>n<br></QueueLowWaterMark> | Specifies the level at which the in-memory file request queue will scan the database for more file requests.<br><br>The default level is 50. |
| <SmallFileSize><br>n<br></SmallFileSize> | Enables packing of small files that are below the specified size into Jar files.<br>The file size may be specified as 'n' bytes or 'nK' for kilobytes. |
| <FilesPerJar><br>n<br></FilesPerJar> | Specifies the maximum number of small files to pack into each Jar file when the Jar packing feature is enabled.<br><br>The SizePerJar setting may also be specified with the FilesPerJar setting.<br><br>The default valus is 25. |
| <SizePerJar><br>n<br></SizePerJar> | Specifies the maximum file size for the Jar file when Jar packing is enabled. The size may be specified as 'n' for bytes, 'nK' for kilobytes or 'nM' for megabytes.<br><br>The FilesPerJar setting may also be specified with the SizePerJar setting.<br><br>The default value is 200K. |
| <JarCompressionLevel><br>n<br></JarCompressionLevel> | Specifies the compression level for the Jar files generated when Jar packing is enabled.<br><br>The valid range is 0 to 9, where 0 is no compression and 9 is the highest compression level. |

| JDBCFileLoader Configuration | |
|---|---|
| | The default value is 0 (no compression). |
| <KeepJars/> | Indicates that the generated Jar files should not be deleted from the temporary cache are after they have been saved by the file loader.<br><br>This setting is useful for testing purposes. |
| <Debug/> | Enables JDBCFileLoader debug output. |

### 2.4.7.2.3 Sample Configurations

This section contains sample filesystem configurations using the database interface and file loader implementations available in the JLAN Server kit.

The following sample configuration uses a Cloudscape/Derby database to hold the filesystem structure details and stores the file data and directory structure on the local filesystem, at N:\DerbyFileSys.

```
<diskshare name="DerbySimple" comment="Derby db filesystem"">
  <driver>
    <class>org.alfresco.jlan.server.filesys.db.DBDiskDriver</class>
    <CacheTime>300</CacheTime>

    <DatabaseInterface>
      <class>org.alfresco.jlan.server.filesys.db.derby.DerbyDBInterface</class>
      <DSN>jdbc:derby:DerbySimpleDB;create=true</DSN>
      <Username>dbuser</Username>
      <Password>dbpass</Password>
      <ConnectionPool>10:20</ConnectionPool>
    </DatabaseInterface>

    <FileLoader>
      <class>org.alfresco.jlan.server.filesys.loader.SimpleFileLoader</class>
      <RootPath>N:\DerbyFileSys</RootPath>
    </FileLoader>
  </driver>
</diskshare>
```

The following sample configuration uses a mySQL database to hold the filesystem structure, load/save queues and file data. The file data is stored using BLOB fields.

The configuration enables the packing of small files into Jar files which are then stored as a single file within the database.

When files are opened the file data will be copied to temporary cache files in the N:\msqltemp\ directory.

```
<diskshare name="MySQLBlob" comment="MySQL virtual filesystem">
  <driver>
    <class>org.alfresco.jlan.server.filesys.db.DBDiskDriver</class>
    <CacheTime>300</CacheTime>

    <DatabaseInterface>
      <class>org.alfresco.jlan.server.filesys.db.mysql.MySQLDBInterface</class>
      <DSN>jdbc:mysql://linuxsrv/JLANNew</DSN>
      <Username>dbuser</Username>
      <Password>dbpassword</Password>
```

```
      <ConnectionPool>10:20</ConnectionPool>
      <FileSystemTable>filesys</FileSystemTable>
      <StreamsTable>filestrm</StreamsTable>
    </DatabaseInterface>

    <FileLoader>
      <class>org.alfresco.jlan.server.filesys.db.DBFileLoader</class>
      <ThreadPoolSize>6:2</ThreadPoolSize>
      <TempDirectory>N:\mysqlcTemp\</TempDirectory>
      <MaximumFilesPerDirectory>1000</MaximumFilesPerDirectory>

      <SmallFileSize>100K</SmallFileSize>
      <FilesPerJar>500</FilesPerJar>
      <SizePerJar>1000K</SizePerJar>
      <JarCompressionLevel>9</JarCompressionLevel>
    </FileLoader>
  </driver>
</diskshare>
```

The following sample configuration uses an Oracle database to hold the filesystem structure, load/save queues and file data. The file data is stored using BLOB fields.

When files are opened the file data will be copied to temporary cache files in the N:\oracleTemp\ directory.

A retention period of seven days will be applied to files/folders created on the filesystem to prevent them from being deleted or modified during the retention period.

The background load/save thread pool will allocate six thread for file loading and two threads for file saving.

```
<diskshare name="OracleBlob" comment="Oracle virtual filesystem using BLOB">
  <driver>
    <class>org.alfresco.jlan.server.filesys.db.DBDiskDriver</class>
    <CacheTime>30</CacheTime>
    <RetentionPeriod>7</RetentionPeriod>

    <DatabaseInterface>
      <class>org.alfresco.jlan.server.filesys.db.oracle.OracleDBInterface</clas
s>
      <DSN>jdbc:oracle:thin:@WIN2000DB:1521:JLAN</DSN>
      <Username>dbuser</Username>
      <Password>dbpassword</Password>
      <ConnectionPool>10:20</ConnectionPool>
    </DatabaseInterface>
```

```
    <FileLoader>
      <class>org.alfresco.jlan.server.filesys.db.DBFileLoader</class>
      <ThreadPoolSize>6:2</ThreadPoolSize>
      <TempDirectory>N:\oracleTemp\</TempDirectory>
      <MaximumFilesPerDirectory>1000</MaximumFilesPerDirectory>
    </FileLoader>
  </driver>
</diskshare>
```

The following sample configuration uses a mySQL database in a clustered configuration to hold the filesystem structure, load/save queues and file data. The file data is stored using BLOB fields.

The configuration enables the packing of small files into Jar files which are then stored as a single file within the database.

When files are opened the file data will be copied to temporary cache files in the N:\msqltemp\ directory. Per node queue and transaction queue table names have been specified as each node must have its own queues for loading/saving file data.

The cluster enabled file state cache is being used so that all nodes in the cluster can co-ordinate access to files, implement cluster wide locking and notify each other of file updates. The near cache is enabled to improve cluster performance. Debug output is enabled to monitor file state cache expiry and file access requests.

```
<diskshare name="MySQLBlob" comment="MySQL virtual filesystem">
  <driver>
    <class>org.alfresco.jlan.server.filesys.db.DBDiskDriver</class>
    <CacheTime>300</CacheTime>

    <DatabaseInterface>
      <class>org.alfresco.jlan.server.filesys.db.mysql.MySQLDBInterface</class>
      <DSN>jdbc:mysql://linuxsrv/JLANNew</DSN>
      <Username>dbuser</Username>
      <Password>dbpassword</Password>
      <ConnectionPool>10:20</ConnectionPool>
      <FileSystemTable>filesys</FileSystemTable>
      <StreamsTable>filestrm</StreamsTable>
      <QueueTable>node1Queue</QueueTable>
      <TransactionQueueTable>node1TransQueue</TransactionQueueTable>
    </DatabaseInterface>

    <FileLoader>
      <class>org.alfresco.jlan.server.filesys.db.DBFileLoader</class>
      <ThreadPoolSize>6:2</ThreadPoolSize>
      <TempDirectory>N:\mysqlcTemp\</TempDirectory>
      <MaximumFilesPerDirectory>1000</MaximumFilesPerDirectory>
```

```
      <SmallFileSize>100K</SmallFileSize>
      <FilesPerJar>500</FilesPerJar>
      <SizePerJar>1000K</SizePerJar>
      <JarCompressionLevel>9</JarCompressionLevel>
    </FileLoader>

    <stateCache type="cluster">
      <clusterName>MySQLCluster</clusterName>
      <clusterTopic>MySQLTopic</clusterTopic>
      <nearCache timeout="10"/>
      <cacheDebug flags="Expire,FileAccess"/>
    </stateCache>
  </driver>
</diskshare>
```

## 2.4.8 Security Configuration

The <*security*> section defines the authentication scheme and access control manager used by the server and defines user accounts. The security section is optional.

The authentication is performed by a class derived from the org.alfresco.jlan.smb.server.Authenticator abstract class. If no authenticator is specified the default authenticator is used that allows access to any user.

The JLAN Server Jar contains two authenticator implementations:-

- org.alfresco.jlan.jlansrv.LocalAuthenticator
  Uses user accounts defined in the configuration file to provide protected access to the virtual filesystems.
- org.alfresco.jlan.jlansrv.PassthruAuthenticator
  Uses a domain controller or other server to authenticate users connecting to the JLAN Server.

Access control is performed by the org.alfresco.jlan.server.auth.acl DefaultAccessControlManager class unless overridden in the configuration.

A global set of access control rules can be specified that are applied to all shares that do not have access control rules assigned.

| **Security Configuration** | |
|---|---|
| <authenticator> | Defines the authentication class to be used and the security mode for the server. |
| <accessControlManager> | Defines the access control manager class, debug output status and additional custom rules. |
| <globalAccessControl> | Defines a set of access control rules to be applied to all shares that do not have access control defined.<br><br>The available access control rules are the same as for a share configuration. See the Shares Configuration section for more details. |
| <JCEProvider>...</JCEProvider> | Configure the JCE provider class.<br><br>For the Cryptix JCE provider specify the provider class as *cryptix.jce.provider.CryptixCrypto.* |
| <users> | Define user accounts for the authenticator class. |

The <*authenticator*> sub-section has the following parameters:-

| **Authenticator Configuration** | |
|---|---|
| <class>...</class> | Specifies the authenticator class. The class must extend the *org.alfresco.jlan.smb.server.Authenticator* abstract class. |
| <mode>..</mode> | Specifies the security mode for the SMB server. Valid values are *USER* or *SHARE*. |

| Authenticator Configuration | |
|---|---|
| <allowGuest/> | Specifies whether unknown users are allowed to access the server. |
| <name>value</name><br>or<br><name/> | Authenticator specific configuration parameters.<br><br>All values in the <*authenticator*> section are passed to the Authenticator *initialize()* method as a NameValueList object. |

The <accessControlManager> sub-section has the following parameters :-

| AccessControlManager Configuration | |
|---|---|
| <class>...</class> | Specifies the access control manager class. The class must implement the *org.alfresco.jlan.server.auth.acl. AccessControlManager* interface.<br><br>If not specified the default access control manager is used (*org.alfresco.jlan.server.auth.acl DefaultAccessControlManager*). |
| <debug/> | Enables access control manager debug output. |
| <rule>...</rule> | Specifies custom access control rule classes to be added to the available access control rule types.<br><br>The value specifies a class that extends the *org.alfresco.jlan.server.auth.acl. AccessControlParser* abstract class.<br><br>If the new access control rule has the same name type as one of the default access control rules it will replace the original rule type.<br><br>Multiple <rule> blocks may be specified. |

A sample <accessControlManager> sub-section is shown below :-

```
<accessControlManager>
  <class>org.alfresco.jlan.server.auth.acl.DefaultAccessControlManager</class>
  <debug/>
  <rule>org.alfresco.jlan.server.auth.acl.DomainAccessControlParser</rule>
</accessControlManager>
```

### 2.4.8.1 LocalAuthenticator

The *org.alfresco.jlan.server.auth.LocalAuthenticator* Authenticator implementation uses a list of users defined in the configuration file to control access to the JLAN

Server virtual filesystems.

The <users> sub-section has the following parameters:-

| **Users Configuration** | |
|---|---|
| <user name=”..”> | Defines a user account on the JLAN Server. |
| <password>..</password> | Defines the password for this user.<br><br>For the *org.alfresco.jlan.server.auth.LocalAuthenticator* class the password is expected to be in plain text. |
| <realname>..</realname> | Specifies the real name of the user for this account. |
| <comment>..</comment> | Comment for this user. The comment is returned by various information requests. |
| <administrator/> | Specifies that the current account is an administrator account. |
| <home>..</home> | Specifies a home directory on the local filesystem for this user.<br><br>When using the default share mapper the user can map to a share called *HOME* that will map to the specified directory. |

A sample security configuration section is shown below:-

```
<security>
  <JCEProvider>cryptix.jce.provider.CryptixCrypto</JCEProvider>
  <authenticator>
    <class>org.alfresco.jlan.server.auth.LocalAuthenticator</class>
    <mode>USER</mode>
    <allowGuest/>
  </authenticator>

  <globalAccessControl default="None">
    <user name="gkspencer" access="Write"/>
  </globalAccessControl>

  <users>
    <user name=”jsmith”>
      <password>mypassword</password>
      <realname>John Smith</realname>
      <comment>Normal user account</comment>
      <home>D:\JohnsFiles</home>
    </user>

    <user name=”admin”>
```

```
      <password>45gHjwm</password>
      <realname>Administrator</realname>
      <administrator/>
    </user>
  </users>
</security>
```

### 2.4.8.2 PassthruAuthenticator

The *org.alfresco.jlan.server.auth.passthru.PassthruAuthenticator* Authenticator implementation uses a domain controller or other network server to authenticate the user connecting to the JLAN Server virtual filesystems.

| **Passthru Authenticator** | |
|---|---|
| <class>...</class> | Specifies the authenticator class. Use *org.alfresco.jlan.server.auth.passthru.PassthruAuthenticator* for the passthru authenticator. |
| <mode>...</mode> | Specifies the security mode. This should be set to USER for the passthru authenticator. |
| <Domain>...</Domain> | Specifies the domain to be used to authenticate users against.<br>The authenticator will search for a domain controller within the specified domain and make a test connection to the selected server. |
| <Server>...</Server> | Name or TCP/IP address of a server to be used to autheticate users against.<br>The authenticator will make a test connection to the server. |
| **<protocolOrder>**<br>**...**<br>**</protocolOrder>** | Specifies the type of protocols and the order of connection for passthru authentication sessions.<br>The default is to use NetBIOS, if that fails then try to connect using native SMB/port 445.<br>Specify either a single protocol type or a comma delimited list with a primary and secondary protocol type.<br>The available protocol types are *NetBIOS* for NetBIOS over TCP and *TCPIP* for native SMB. |
| **<offlineCheckInterval>**<br>**...**<br>**</offlineCheckInterval>** | Specifies how often passthru servers that are marked as offline are checked to see if they are now online.<br>The default check interval is 5 minutes. The check interval is specified in seconds. |
| **<Timeout>**<br>**...**<br>**</Timeout>** | Sets the socket timeout used when connecting a new session to a passthru authentication server.<br>The default timeout value is 5 seconds. The timeout value is specified in seconds. |

A sample security configuration section is shown below:-

```
<security>
```

```
    <authenticator>
      <class>
        org.alfresco.jlan.server.auth.passthru.PassthruAuthenticator
      </class>
      <mode>USER</mode>
      <Domain>STARLASOFT</Domain>
    </authenticator>
  </security>
```

### 2.4.8.3 Enterprise Authenticator

The *org.alfresco.jlan.server.auth.EnterpriseCifsAuthenticator* Authenticator implementation provides support for newer CIFS authentication types such as NTLMSSP, SPNEGO, NTLMv2 and Active Directory/Kerberos.

| **Enterprise CIFS Authenticator** | |
|---|---|
| <class>...</class> | Specifies the authenticator class. Use *org.alfresco.jlan.server.auth.EnterpriseCifsAuthenticator* for the enterprise authenticator. |
| <mode>...</mode> | Specifies the security mode. This should be set to USER for the passthru authenticator. |
| <KDC>...</KDC> | IP address or DNS name of the Active Directory server. |
| <Realm>...</Realm> | Kerberos realm. |
| <Password>...</Password> | Account password used by the server to get a service ticket. |
| <LoginEntry>...</LoginEntry> | Java security login configuration file entry name.<br><br>Defaults is 'JLANServerCIFS'. |
| <disallowNTLMV1/> | Do not allow weaker NTLMv1 logins. |
| **<kerberosDebug/>** | Enables Java API debug output.<br>Using this setting is equivalent to setting the system properties *sun.security.jgss.debug* and *sun.security.krb5.debug* to true. |

A sample security configuration section is shown below:-

```
  <security>
    <authenticator>
      <class>
        org.alfresco.jlan.server.auth.EnterpriseCifsAuthenticator
      </class>
      <KDC>win2003.alfresco.com</KDC>
      <Realm>ALFRESCO</Realm>
      <Password>password</Password>
```

```
    </authenticator>
  </security>
```

See the **Enterprise Authentication Setup** section for more details on how to configure an Active Directory account for use by the JLAN CIFS Server.

### 2.4.9 Share Mapper Configuration

The <*shareMapper*> section defines the custom share mapper interface.

| **ShareMapper Configuration** | |
|---|---|
| <class>..</class> | Specifies the share mapper class. The class must implement the *org.alfresco.jlan.smb.server.ShareMapper* interface. |
| <debug/> | Enable debug output for the share mapper. |
| <name>value</name><br>or<br><name/> | ShareMapper specific configuration parameters.<br><br>All values in the <*shareMapper*> section are passed to the ShareMapper *initializeMapper()* method as a NameValueList object. |

A sample share mapper configuration section is shown below:-

```
<shareMapper>
  <class>org.alfresco.jlan.smb.server.DefaultShareMapper</class>
  <debug/>
</shareMapper>
```

### 2.4.10 Drive Mappings Configuration

The <DriveMappings> section defines local drive mappings that will be added when the JLAN Server SMB/CIFS server starts. This can be useful when using the JLAN Server to provide custom filesystems to the local host.

The drives are mapped after the SMB/CIFS server component has started and removed as the SMB/CIFS server shuts down.

| **DriveMappings Configuration** | |
|---|---|
| **<mapDrive drive=".." share="..." [username="..."] [password="..."] [interactive="yes\|no"] [prompt="yes\|no"] />** | Creates a mapped local drive to the specified JLAN Server shared filesystem.<br>The drive parameter specifies the name of the local mapped drive to create, such as Z:.<br>The share parameter specifies the name of the JLAN Server share to map the drive to. The share must be defined in the <shares> section of the configuration.<br>The username and password are optional parameters that specify the logon details when connecting the mapped drive. If not specified the default credentials are used. |

| DriveMappings Configuration | |
|---|---|
| | The interactive parameter specifies whether a dialog is displayed to prompt for a username and password if the default or supplied credentials cannot logon to the JLAN Server.<br><br>The prompt parameter specifies that the username/password dialog is displayed before the connection attempt is made. |

## 2.4.11 Debug Configuration

The <*debug*> section defines the debug output handler, the debug section is optional.

| Debug Configuration | |
|---|---|
| <output> | Specifies the debug output class and debug class specific configuration parameters. |

| Output Configuration | |
|---|---|
| <class>..</class> | Specifies the debug output class to be used. The class must implement the *org.alfresco.jlan.debug.DebugInterface* interface.<br><br>The JLAN Server Jar contains four debug output classes. The *org.alfresco.jlan.debug.ConsoleDebug* class outputs all debug information to the console.<br><br>The *org.alfresco.jlan.debug.LogFileDebug* class outputs all debug information to a file.<br><br>The *org.alfresco.debug.JDKLoggingDebug* class outputs all debug information using the JDK Logging APIs.<br><br>The *org.alfresco.debug.cluster.ClusterDebug* class sends all debug output to other nodes within the cluster, as well as logging locally using one of the above debug output classes. |
| <name>value</name><br><br>or<br><br><name/> | Debug specific configuration parameters.<br><br>All values in the <*output*> section are passed to the Debug *initialize()* method as a NameValueList object.<br><br>The *org.alfresco.jlan.debug.ConsoleDebug* class does not have any initialization parameters.<br><br>The *org.alfresco.jlan.debug.LogFileDebug* class has a <*logFile*>..</*logFile*> parameter to specify the output log file name, and a <*append/*> parameter to specify that output is appended to the log file on each run of the server. |

| Output Configuration | |
|---|---|
| | The *org.alfresco.jlan.debug.JDKLoggingDebug* class has a *<Properties>..</Properties>* paramter to specify the JDK logging properties file. |

A sample debug configuration section is shown below:-

```
<debug>
  <output>
    <class>org.alfresco.jlan.debug.LogFileDebug</class>
    <logFile>jlansrv.log</logFile>
    <append/>
  </output>
</debug>
```

### 2.4.11.1 Cluster Debug Configuration

When running with filesystems that are using a cluster configuration, or to send output from multiple standalone JLAN Servers to a central logging server, the ClusterDebug class is used.

The cluster debug implementation uses the Hazelcast code to provide the underlying cluster connection with other nodes. The Hazelcast instance is shared between the debug interface and any filesystems that are configured to use the clustered file state cache.

The node which is to act as the central logging server for the cluster is configured to be a receive only node for the debug interface. This node does not broadcast its debug output to the cluster.

The cluster debug interface sends a copy of the debug output to the cluster and passes a copy to another debug interface to handle the local output. This may be the console, file or JDK logging debug interface or a custom debug interface of your own.

| Cluster Debug Configuration | |
|---|---|
| <class>...</class> | Use the *org.alfresco.debug.cluster.ClusterDebug* class |
| <debugTopic><br>…<br></debugTopic> | Hazelcast topic that debug output messages will be sent to. All nodes in the same cluster must use the same topic name. |
| <receiveOnly/> | Specifies that this is the node that will log debug messages from all other nodes in the cluster.<br>This node will not broadcast debug output to the cluster. |
| <localOutput> | Configures the debug interface to be used for local output.<br>Either *org.alfrsco.debug.ConsoleDebug*, *org.alfresco.debug.LogFileDebug* or *org.alfresco.debug.JDKLoggingDebug* should be used, or a custom debug interface of your own. |

A sample cluster debug configuration section is shown below, sending local output to the console, and acting as the central debug logger for the cluster:-

```
<debug>
  <output>
    <class>org.alfresco.jlan.debug.cluster.ClusterDebug</class>
    <debugTopic>AlfrescoJLANDebug</debugTopic>
    <receiveOnly/>

    <localOutput>
      <class>org.alfresco.jlan.debug.LogFileDebug</class>
      <logFile>jlansrv.log</logFile>
      <append/>
    </localOutput>
  </output>
</debug>
```

# 3 Deploying The SMB/CIFS Server On Windows

The Windows SMB/CIFS server is integrated into the core Windows networking services and is usually enabled by default, to allow access to the IPC$ named pipe and admin shares (C$, D$ etc.).

Windows uses two underlying protocols to access the SMB/CIFS server, native SMB and NetBIOS over TCP/IP. There are other protocols used such as NetBEUI and IPX but as Java only supports TCP/IP in the core libraries we only discuss the TCP/IP based protocols here.

## 3.1 Windows Native SMB/CIFS

Native SMB/CIFS sessions use a TCP/IP socket connection to port 445 on the file server. Windows binds port 445 globally, ie. it is bound to all available network adapters including the localhost address of 127.0.0.1.

Native SMB/CIFS is only available on Win2000 and later versions of Windows.

Windows use of native SMB/CIFS can be controlled via the following registry key :-

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NetBT\Parameters] "SMBDeviceEnabled"=dword:00000001
```

Native SMB/CIFS is enabled by default. Changing the value to zero and rebooting the system will disable native SMB/CIFS support.

The native SMB/CIFS service is designed to use DNS to lookup host names.

## 3.2 Windows NetBIOS Over TCP/IP

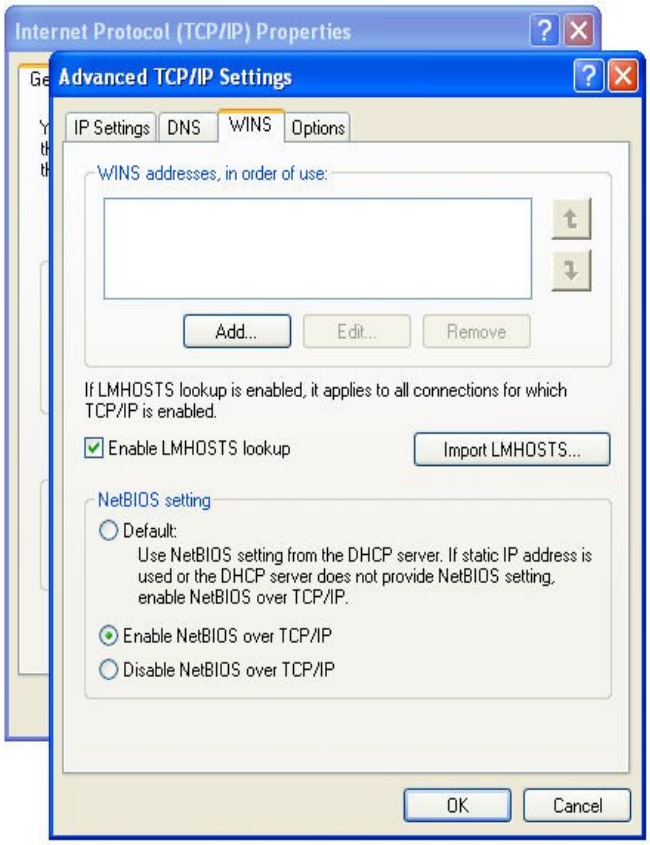
NetBIOS over TCP uses a combination of TCP sockets and UDP datagrams. A session is established using a TCP/IP socket connection to port 139 on the file server. The NetBIOS over TCP/IP protocol binds on a per network adapter basis.

Datagrams may be used to send/receive various announcements such as name registrations, name lookup requests, host announcements. Ports 137 and 138 are used to send/receive datagrams.

NetBIOS over TCP/IP contains it's own naming support. Name lookups may be configured to use a network broadcast or use a name server (WINS), or a combination of both.

NetBIOS host names may also be specified in the LMHOSTS local file that is usually located in the *<Windows>\system32\drivers\etc* directory. This directory often contains a sample LMHOSTS file named *LMHOSTS.SAM*, the file extension may be hidden depending on your Windows Explorer settings.

The Network Connection Properties dialog can be used to enable/disable NetBIOS over TCP/IP.




Windows does not bind NetBIOS over TCP/IP to the localhost/127.0.0.1 address.

## 3.3 JLAN Server SMB/CIFS Implementation

The JLAN Server implements native SMB/CIFS, NetBIOS over TCP/IP and can also use the Windows NetBIOS over TCP/IP code as the session and transport layer via the Win32 Netbios() API call.

The following sections detail how to configure the various protocol implementations when running under Windows.

### 3.3.1 Native SMB/CIFS

To use the JLAN Server native SMB/CIFS implementation under Windows requires that the Windows native SMB/CIFS server be disabled. The JLAN Server kit includes the *port445.reg* registry file which can apply the relevant change to disable the Windows natvie SMB/CIFS server, a reboot is required after applying this change.

The followng SMB/CIFS XML configuration section is the minimum configuration required to enable JLAN Server native SMB/CIFS support :-

```
<SMB>
  <host name="JLANSRV" domain="ALFRESCO">
    <tcpipSMB/>
  </host>
</SMB>
```

The JLAN Server native SMB/CIFS session handler may also be bound to a particular network address :-

```
<SMB>
  <host name="JLANSRV" domain="ALFRESCO">
    <tcpipSMB/>
    <bindto>192.168.1.2</bindto>
```

```
  </host>
</SMB>
```

### 3.3.2 NetBIOS Over TCP/IP

To use the JLAN Server NetBIOS over TCP/IP SMB/CIFS implementation under Windows requires that either the Windows TCP/IP NetBIOS support is disabled on a network adapter or the localhost/127.0.0.1 address is used. The localhost/127.0.0.1 address is only useful if you are using the JLAN Server to provide a filesystem that is accessible locally.

TCP/IP NetBIOS support requires that the network broadcast mask be specified in the configuration. The broadcast mask can be determined from the network mask, for example if the network mask is 192.168.1.0 the broadcast mask is 192.168.1.255. A broadcast mask of 255.255.255.255 can be specified, but a specific broadcast mask should be used where possible.

The following SMB/CIFS XML configuration section is the minimum configuration required to enable JLAN Server NetBIOS over TCP/IP support :-

```
<SMB>
  <host name="JLANSRV" domain="ALFRESCO">
    <netBIOSSMB/>
    <broadcast>192.168.1.255</broadcast>
  </host>
</SMB>
```

The above configuration will bind globally to all available network adapters. The network adapter that the NetBIOS over TCP/IP handler binds to can also be specified :-

```
<SMB>
  <host name="JLANSRV" domain="ALFRESCO">
    <netBIOSSMB/>
    <broadcast>192.168.1.255</broadcast>
    <bindto>192.168.1.2</bindto>
  </host>
</SMB>
```

To have the JLAN Server appear under Network Neighborhood the host announcer must be enabled :-

```
<SMB>
  <host name="JLANSRV" domain="ALFRESCO">
    <netBIOSSMB/>
    <broadcast>192.168.1.255</broadcast>
    <bindto>192.168.1.2</bindto>
    <hostAnnounce interval="5"/>
  </host>
</SMB>
```

To use the JLAN Server to provide virtual filesystems to the localhost only you can bind the JLAN Server to the localhost/127.0.0.1 address :-

```
<SMB>
  <host name="JLANSRV" domain="ALFRESCO">
    <netBIOSSMB bindto="127.0.0.1"/>
    <broadcast>192.168.1.255</broadcast>
    <bindto>127.0.0.1</bindto>
  </host>
</SMB>
```

Enabling the host announcer in this configuration has no effect, instead an entry should be added to the local LMHOSTS file in the *<Windows>\system32\drivers\etc* directory.

The localhost/127.0.0.1 configuration does not work under WinXP SP2 due to security changes introduced by Microsoft. See the Win32 NetBIOS solution below.

If your network is configured to use WINS for NetBIOS naming use the following configuration :-

```
<SMB>
  <host name="JLANSRV" domain="ALFRESCO">
    <netBIOSSMB/>
    <broadcast>192.168.1.255</broadcast>
    <WINS>
      <primary>192.168.1.10</primary>
    </WINS>
  </host>
</SMB>
```

### 3.3.3 Win32 NetBIOS

The Win32 Netbios API allows the JLAN Server code to use the Windows NetBIOS over TCP/IP code to publish the server name, accept incoming sessions and announce the server to Network Neighborhood.

Using the Win32 Netbios API allows the JLAN Server to co-exist with the Windows file server without requiring changes to the network configuration. The JLAN Server is accessible via the network and from the localhost.

The following SMB/CIFS XML configuration section is the minimum configuration required to enable JLAN Server Win32 NetBIOS support :-

```
<SMB>
  <host name="JLANSRV" domain="ALFRESCO">
    <Win32NetBIOS/>
  </host>
</SMB>
```

If there are multiple network adapters in the server system you may need to specify the NetBIOS logical adapter, known as a *LANA.*The easiest way to determine the available LANAs is to enable the 'Socket' debug level using the <sessionDebug flags="..."/> setting within the <SMB> configuration section, this will dump out a list of the available LANAs.

The following configuration section shows how to use a particular LANA for the Win32 NetBIOS interface :-

```
<SMB>
  <host name="JLANSRV" domain="ALFRESCO">
    <Win32NetBIOS lana="6"/>
  </host>
</SMB>
```

To configure the Win32 NetBIOS interface so that the JLAN Server is only accessible to a particular host the *accept* configuration attribute is used :-

```
<SMB>
  <host name="JLANSRV" domain="ALFRESCO">
    <Win32NetBIOS accept="MYPC"/>
  </host>
</SMB>
```

With the above configuration the JLAN Server Win32 NetBIOS interface will only accept connections from the host with a NetBIOS name of *MYPC,* this could be the localhost or a network host.

To have the JLAN Server appear under Network Neighborhood when using the Win32 NetBIOS interface use the following configuration settings :-

```
<SMB>
  <host name="JLANSRV" domain="ALFRESCO">
    <Win32NetBIOS/>
    <Win32Announce interval="5"/>
  </host>
</SMB>
```

# 4 Enterprise Authentication Setup

The JLAN Server Jar includes an authenticator component that implements the newer CIFS authentication methods such as SPNEGO, NTLMSSP, NTLMv2 and Kerberos/Active Directory.

In order to use the Kerberos/Active Directory integration extra steps are required to setup the service ticket for the JLAN Server CIFS server.

See section 2.4.7.3 for details of the options available when configuring the JLAN Server to use the Enterprise authenticator.

## 4.1 Kerberos/Active Directory Setup

This section details how to setup an account under Active Directory for use by the JLAN CIFS server.

1. Create a user account for use by the JLAN CIFS server using the Active Directory Users and Computers application. Use the Action->New->User menu, then enter the full name, such as 'JLAN Server CIFS', and the user logon name, such as 'jlanservercifs'. Click Next, enter a password, enable 'Password never expires' and disable 'User must change password at next logon'. Click Finish.

   Right click the new user account name, select Properties, go to the Account tab and enable the *Use DES Encryption types for this account* and *Do not require Kerberos preauthentication* options in the Account Options section.

2. Use the *ktpass* utility to generate a key table. The *ktpass* utility is a free download from the Microsoft site, and is also part of the Win2003 Resource Kit. There is a restriction in the Microsoft *ktpass* utility, use this commandline on the Domain Controller only :-

   *ktpass -princ cifs/<cifs-server-name>.<domain>@<realm>*
   *-pass <password> -mapuser <domainnetbios>\jlanservercifs -crypto DES-CBC-MD5*
   *-ptype KRB5_NT_PRINCIPAL -mapop set +desonly*
   *-out c:\temp\jlanservercifs.keytab*

   The principal should be specifed using the server name and domain in lowercase with the realm in uppercase.

   **Note: Some versions of the ktpass command can generate invalid keytab files, download the latest version from the Microsoft web site to avoid any problems.**

3. Create the Service Principal Names (SPN) for the JLAN Server CIFS server using the *setspn* utility. The *setspn* utility is a free download from the Microsoft site, and is also part of the Win2003 Resource Kit.

   *setspn -a cifs/<cifs-server-name> jlanservercifs*
   *setspn -a cifs/<cifs-server-name>.<domain> jlanservercifs*

   You can list the SPNs for an account using :-

   *setspn -l jlanservercifs*

   *<cifs-server-name>* is the NetBIOS name of the Alfresco CIFS server when running on

an Active Directory client or the host name for a client that is not an Active Directory client, ie. not logged onto the domain.

Some versions of the *ktpass* command will add the SPN for the principal so you may only need to add the NetBIOS/short name versions of the SPNs. Use the *setspn -l <account-name>* command to check if the *ktpass* command set the SPN.

4. Copy the *cifs.keytab* file to the server where the JLAN Server will run. Copy the file to a protected are such as C:\etc\ or /etc.
5. Setup the Kerberos ini file on the server that the JLAN Server will run, the default location is C:\winnt\krb5.ini or /etc/krb5.conf. A sample krb5.ini is shown below.

```
[libdefaults]
default-realm = ALFRESCO.ORG

[realms]
ALFRESCO.ORG = {
  kdc = adsrv.alfresco.org
  admin-server = adsrv.alfresco.org
}

[domain-realm]
adsrv.alfresco.org = ALFRESCO.ORG
.adsrv.alfresco.org = ALFRESCO.ORG
```

Note: The realm should be specified in uppercase.

6. Setup the Java login configuration file. This would usually be in the *JRE\lib\security* folder. Create a file named *jlan.login.config* with the following entry :-

```
JLANServerCIFS {
 com.sun.security.auth.module.Krb5LoginModule required
 storeKey=true
 useKeyTab=true
 keytab="C:/etc/cifs.keytab"
 principal="cifs/<jlan-server-name>.<domain>";
};
```

7. Enable the login config file in the main Java security configuration file, usually at *JRE\lib\security\java.security*. Add the following line :-

```
login.config.url.1=file:${java.home}/lib/security/jlan.login.config
```

8. Configure the JLAN CIFS server to use the Enterprise authenticator with Kerberos enabled :-

```
<authenticator>
 <class>
 org.alfresco.jlan.server.auth.EnterpriseCifsAuthenticator
 </class>
 <mode>USER</mode>
 <allowGuest/>
 <Debug/>
 <KDC>adsrv.starlasoft.co.uk</KDC>
 <Realm>STARLASOFT.CO.UK</Realm>
 <Password>...</Password>
 <Principal>cifs/<cifs-server-name>.<domain></Principal>
</authenticator>
```

Where <Password> is the account password from step 1.

To help diagnose problems with the Kerberos/Active Directory setup you can enable debug output from the Java security APIs by defining the following property on the command line of the JVM :-

-Dsun.security.krb5.debug=true