ConnectedWay™, LLC

Authentication Mechanisms for ConnectedNAS™ SMB Client

White Paper

Date: October 2019

File Sharing for All Devices

Preface

ConnectedNAS™ is a powerful and robust software stack that enables remote file sharing using the standard "Server Message Block", SMBv1/v2/v3, networking protocols. These protocols are in use today in Microsoft® Windows® products, MacOS®, and Linux systems using the SAMBA software suite. The protocol is also used by various embedded and OEM products.

ConnectedNAS® is designed for the OEM and mobile markets. It is supported by various commercial RTOSes and deployed on a range of architectures. ConnectedNAS supports both client and server use cases. Client use cases involve accessing remote files on a deployed system while Server uses cases involve sharing local files with remote computers.

Given the robust nature of the SMB protocol and the types of systems that it has been deployed on, various mechanisms for authenticating access have arisen. Some mechanisms are appropriate for home or small office environments while others are more suited for enterprise or cloud environments.

This paper will discuss supported models and use cases for authenticating client access to remote files.

ConnectedWay™, LLC

Authentication Mechanisms for ConnectedNAS™ SMB Client

October 2019

Version 1.0 Final

Prepared by

Richard Schmitt

rschmitt@connectedway.com

Owner and Product Architect

Table of Contents

Executive Summary

ConnectedNAS™ is a robust, professional, and commercial software package that enables networked file sharing between client and server devices. In order to ensure privacy and security of networked content, it is imperative that access be authenticated.

In the early days of networking, an identity could be established by simply exchanging a username and password in cleartext between partners. Over the course of the following decades, more sophisticated mechanisms for establishing this identity have evolved.

Today, there are essentially two approaches for authenticating a user: one where a user is authenticated against a specific computer or device and another where the user is authenticated on the network and that authentication is used to access resident computers and devices.

There are various ways these two approaches can be deployed in a system to provide various capabilities and use cases. The architecture of the ConnectedNAS™ authentication mechanisms is discussed and supported use cases for deployment will be detailed.

Terminology

When discussing authentication, there are concepts and terms which are often confusing or overloaded with other terms, making an understanding of the concepts difficult. To help with this challenge, the terms used throughout this document are introduced here. Many of these concepts and terms will be discussed in more detail in follow-up sections.

Term	Description
Active Directory	A Microsoft® term used to define the system that authenticates users on a network.
Asymmetric Key	A public or private key used as part of a key pair. One endpoint uses the public key while the other endpoint uses the private key. An asymmetric key pair is useful to provide a remote with a public encryption key that can only be decrypted with the private key.
Authentication	The act of validating that a particular user has access to some resource
Authid	A GSSAPI term which represents the authenticated object. This is synonymous with security principal.
Confidentiality	A capability that the endpoints of a communications channel can provide whereby the data being transferred cannot be read by any non-participating node. (i.e. encryption)
Credential	Metadata used to authenticate with a File Server or a Directory Server
Credential Cache	An active directory term referring to a set of authentication credentials that are currently active. The Kerberos stack typically manages a credential cache.
Directory Server	A server used for authentication when Domain Mode is used.
Domain	A collection of users and resources managed by a Microsoft® Windows Active Directory.
Domain Mode	A ConnectedNAS™ concept that specifies that authentication is performed against a network rather than an individual server
File Server	A server that allows access to locally sourced files. Distinct from a "Directory" Server, and a "Doman Controller" although the latter two are also sometimes called servers
FQDN	A "Fully Qualified Domain Name". Directory Servers are specified with a FQDN. An FQDN is a DNS name such as directory.example.com. Fully Qualified implies that it is complete, containing no default or implied names.
GSSAPI	An application API used by ConnectedNAS™ to access the authentication libraries.

Integrity	A capability that the endpoints of a communications channel can provide whereby the data being transferred can be guaranteed not to be modified by any non-participating node. (i.e. signing)
Kerberos	A technology and protocol used for network authentication
Key	A cryptographic blob used to access data. A key is specific to the cryptographic algorithms being used. There are two basic forms of keys: symmetric and asymmetric.
NTLMSSP	Also simply called NTLM, it is an authentication protocol that allows file server sign-on (as opposed to network sign-on).
Password	A sequence of bytes, typically a word or phrase, used to establish a trust relationship between two systems. Passwords should never be stored in cleartext or transferred over a network.
Principal	Also known as a Security Principal. An authenticated object. A fully specified user is a principal as well as a computer or even a service.
Realm	The set of authentication entities managed by a directory server. Similar to a Microsoft® Windows Domain.
Samba	An open source software package that provides remote file access and other functions. Primarily used on Linux although available on other platforms.
Samba ad-dc	The Active Directory/Domain Controller component of the Samba open source solution.
SASL	Secure Authentication API and Programming paradigm. SASL allows a single API that interfaces to multiple and nested authentication mechanisms. ConnectedNAS™ utilizes a SASL programming model.
Server Mode	Synonymous with Workgroup Mode
Session	A state where a user has authenticated. Authentication can have been acquired in Domain Mode or Workgroup Mode.
Session Key	A key derived from authentication and never transferred directly between endpoints. Used to provide confidentiality and integrity of the data transfer during the session.
Single-Sign-On	An authentication process synonymous with Domain Mode that allows a single sign on to the network, and that subsequent access to other file servers within the network can utilize the previous authentication.
SPNEGO	A protocol wrapper for authentication protocols that allow for negotiation of the authentication mechanism by the end points.

Symmetric Key	A key shared between two endpoints that can encrypt and decrypt a data blob.
Username	Represents a user, but it is not fully qualified. To be useful with Kerberos, it would need to be further specified with a FQDN.
Workgroup Mode	A ConnectedNAS™ concept that specifies that authentication is performed on a server device rather than a network. Also referred to as Server Mode

Authentication Mechanisms

Authentication is the process of proving that a client is who they say they are and that that client has the appropriate privileges for accessing the server. The simplest form of authentication is a username/password. In this scheme, the server knows all possible users and their respective passwords. If a client can present the username and correct password to the server, then that user is able to authenticate.

There are numerous problems with this approach. The most glaring is that if a bad actor is able to snoop on the network, they would be able to observe the presented username and password. They would then be free to use those credentials for obtaining their own access at some later time. This type of access is called cleartext authentication and it is no longer used in smbv2/v3.

An authentication mechanism used by some secure websites leverages public key encryption. This mechanism uses the concept of asymmetric and symmetric keys to encrypt an authentication session. The client contacts the server and the server in turn generates a public/private key pair. The server will keep the private key and send the public key to the client. The client creates a symmetric session key, encrypts it using the public key and sends it to the server. Since the server has the private key, it is able to decrypt the session key. It then uses the session key to encrypt and decrypt the authentication session. This type of authentication is called PKI and is not used in SMB file sharing.

Early versions of SMB used an authentication mechanism called LM involving what is called shared secret. In this type of authentication, each side knows the password, but it is never transferred over the network. Instead, the password is used to create a hash, or secret, using the same algorithm on both the client and the server. The secret is shared and if it matches, it proves that the client knew the correct password in order to generate the secret. This type of authentication has numerous weaknesses and is also not used.

A variation on shared secret is challenge response. With challenge response, the client notifies the server that it wishes to authenticate. The server creates a unique challenge and sends that to the client. The server uses this challenge along with the password to create a unique response. Since the server remembers the challenge, it knows what the response should look like if the client knows the correct password. This is the fundamental approach to NTLMSSP, the authentication mechanism used in the SMB Workgroup model.

The last type of authentication we will discuss is called Kerberos, which is a ticket based, third party mechanism. The inner workings of Kerberos utilize a combination of asymmetric and symmetric keys to prove the identity of the client to the server. Because it involves a third party, the ticket can be shared across the network to prove to multiple parties that the client knows the password. This mechanism is the authentication used by Active Directory, the Domain model of ConnectedNAS™.

In summary, ConnectedNAS™ utilizes either NTLM or Kerberos. Which mechanism is used depends on many factors. Primarily, it depends on the configuration of the network and servers that the client is deployed within. Secondarily, it depends on the configuration and build of the ConnectedNAS™ SMB client. To use Kerberos, ConnectedNAS™ requires a Kerberos implementation to interface with on the client. ConnectedNAS™ supports MIT Kerberos-5 and provides packaging for this on many of our supported platforms.

SMB Protocol

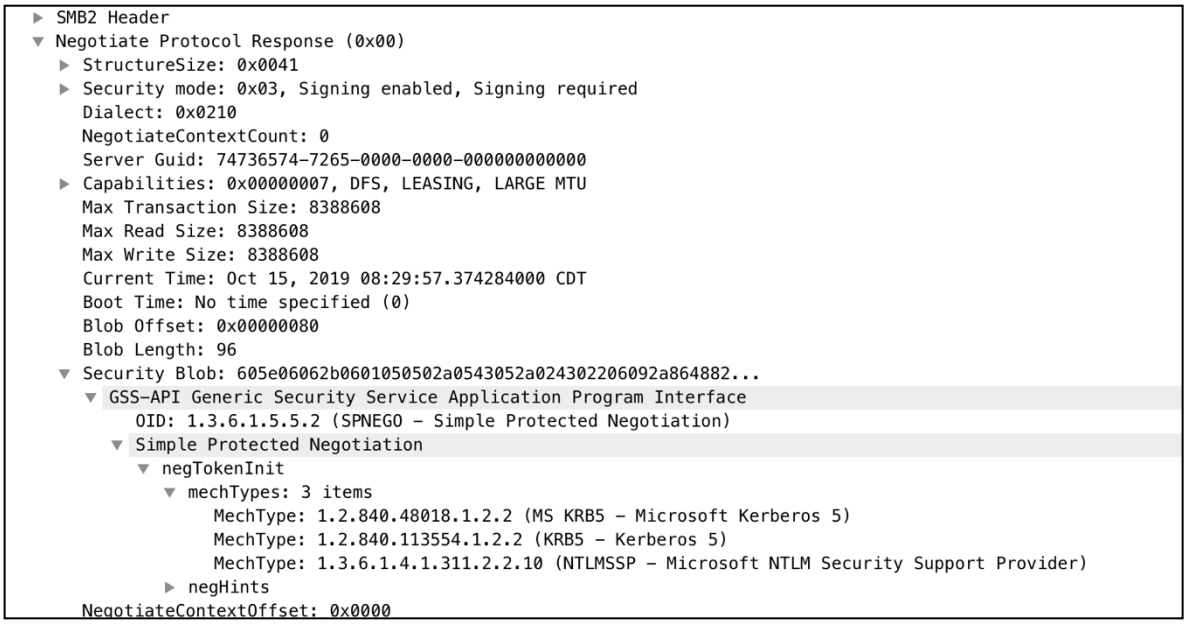
Early versions of the SMB protocol have been around since the advent of personal computers. It has undergone numerous enhancements and one major revision (although the version numbering implies 2 major revisions). SMBv1 was part of the early LANMGR support and only now is being deprecated. SMBv1 had numerous security vulnerabilities, its resource management was cumbersome, inefficient, and complex yet limited for its support of enhanced functionality. SMBv1 was also called CIFS. In 2006, SMBv2, a rewrite of the underlying protocol, was released. SMBv3 is considered an incremental update to the v2 version primarily focused on functional enhancements for virtualization and security.

ConnectedNAS™ supports all versions but our SMBv1 stack is being deprecated and our SMBv3 support is limited to SMBv2 functionality with the addition of SMBv3 encryption.

SMB is a client / server protocol. The file request initiator is the client and the remote device that hosts the desired content is the server. The protocol is session based. Establishing a session involves setting up a network connection, negotiating the SMB protocol, authenticating and setting up security for the session. Once a session is established, the client can connect to shares, and perform file operations.

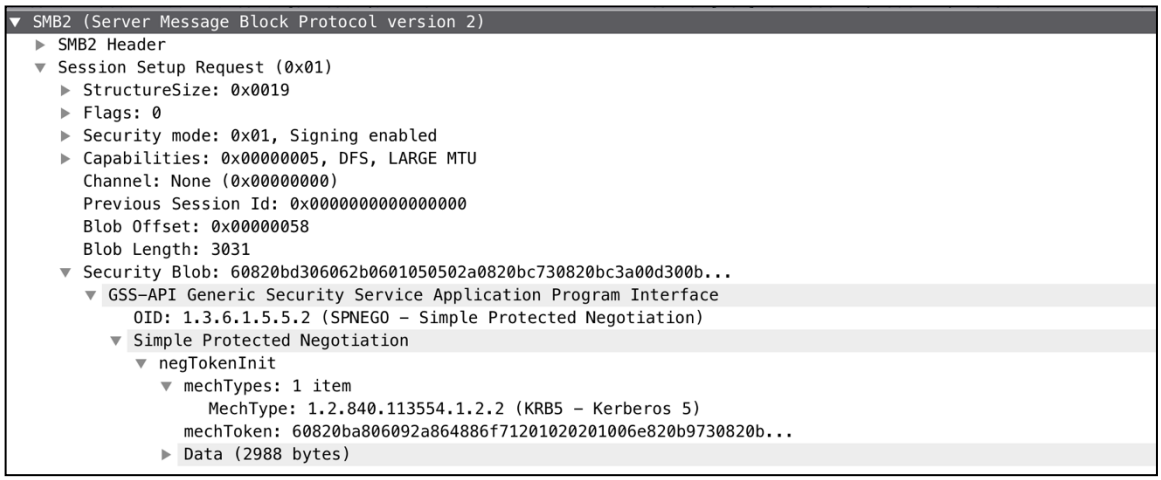
This paper will address how ConnectedNAS™ negotiates the authentication mechanisms, executing the mechanism, and using data from the authentication to provide confidentiality and integrity throughout the session.

The first protocol exchange is the negotiation exchange. An example of a decoded negotiation packet is shown below:



From this packet, it is clear that the server supports three types of authentication mechanisms: Microsoft® Kerberos 5, Generic Kerberos 5, and NTLMSSP. From this information, the client is able to determine the SMB version and the authentication mechanism to use. The second stage is the session setup stage. During session setup, the client may enter into a negotiation with a third party to obtain an authentication ticket and present this ticket as part of the setup stage (as is the case with Kerberos), or may negotiate directly with the server (as is the case with NTLMSSP).

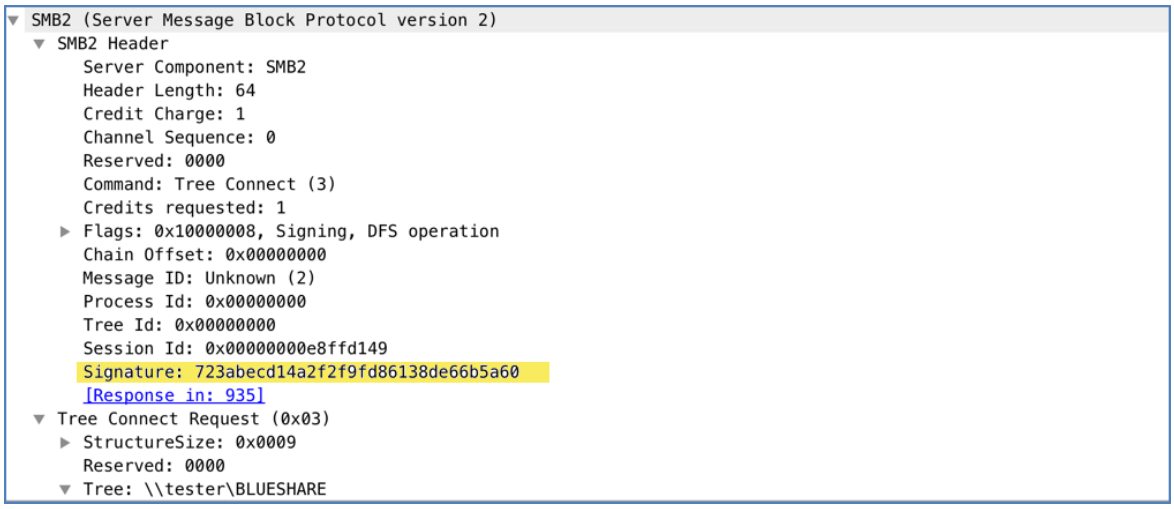
An example of the Session Setup Request is shown below. The client has chosen KRB5 as the authentication mechanism and has presented the ticket to the server.



The actual authentication mechanism used is independent of the SMB protocol. It is specified by the GSSAPI and SPNEGO headers. The authentication protocol and data is embedded as an opaque blob into the setup packets. As a matter of practicality, all modern SMB servers support either Kerberos or NTLMSSP or both. Since ConnectedNAS™ supports both mechanisms, it is able to authenticate with any SMB server.

The authentication negotiation also produces a "session key". As is the case with a shared secret, the session key is never actually sent across the network, but rather it is derived by both the client and server. This session key is used to calculate a signature over the SMB protocol packets and placed within the SMB protocol header thereby ensuring the integrity of the protocol packet.

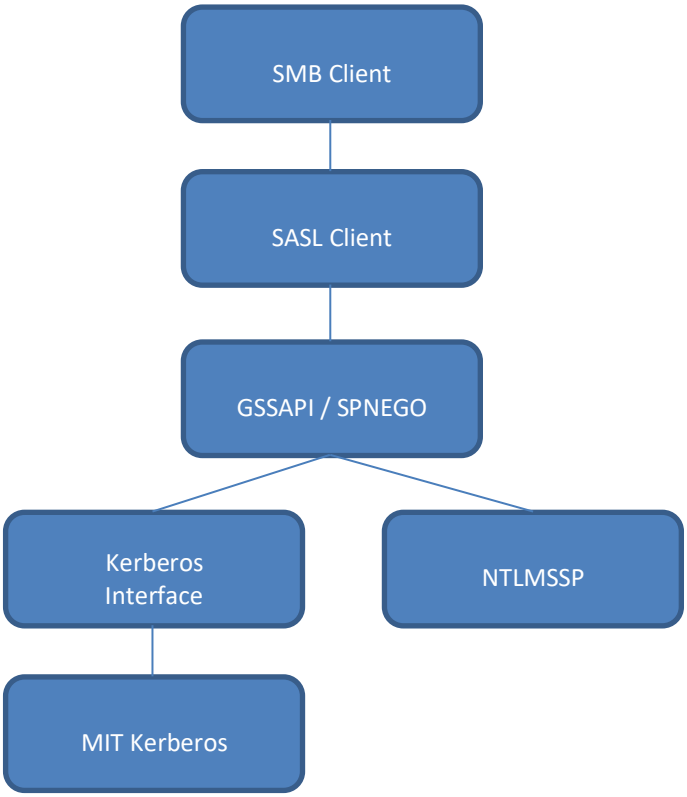
Following the session setup, subsequent SMB packets will contain a signature in the header as shown below:



A bad actor on the network is unable to modify the SMB packets since it is not privy to the session key and cannot regenerate a valid packet signature. A derivation of the session key can also be used by the endpoints to encrypt and decrypt the protocol packets and data.

ConnectedNAS™ Security Architecture

A high-level architectural view of the ConnectedNAS™ Security Architecture is shown below:



ConnectedNAS™ includes a SASL software library derived from Cyrus SASL. We've written our own GSSAPI/SPNEGO layer that plugs into the SASL stack. Although, as shown in the packet contents above, GSSAPI and SPNEGO are distinct protocols, we have implemented them as a single protocol layer since they will always appear together. Our GSSAPI layer supports two authentication protocols: Kerberos and NTLMSSP. The plugin API for GSSAPI, Kerberos, and NTLMSSP are all compatible. Technically, our Kerberos and NTLMSSP layers could connect directly to our SASL layer but that would remove the negotiation ability of the SPNEGO layer.

The Kerberos plugin for SASL interfaces with the third-party MIT Kerberos Library. This implies that in order to support the Kerberos authentication mechanism, the MIT Kerberos library must be installed or ported to the platform. ConnectedWay has ported the MIT Kerberos library onto Android and integrated it into our ConnectedNAS™ Storage Provider for Android. We have also integrated ConnectedNAS™ with the MIT Kerberos library on MacOS, Linux, and other platforms. Please discuss your specific RTOS requirements with ConnectedWay™ to understand the availability of Kerberos on your platform.

SASL

ConnectedNAS™ chose a security architecture based on the "Simply Authentication and Security Layer (SASL) specified as RFC 2222. Alternatives to SASL could have been Microsoft® SSPI, Samba's GSE, or a proprietary architecture. We derived our SASL layer from the Cyrus SASL package. Cyrus SASL is licensed under the permissive CMU software license which allows use, modification, and distribution in source or binary form as long as the origin of the software is noted within the code and in the distribution.

There are some limitations to the Cyrus SASL library as distributed. It's support for SMB is limited and does not support session keys. So, we have ported and rewritten parts of this library.

SASL presents a fairly simply API implemented by most security providers. SASL is session oriented. There are calls to create sessions and start an authentication exchange within the session. The authentication exchange involves successive calls to a step function. Where opaque blobs are passed in and out of the SASL layer. These opaque blobs are simply copied as is into and out of the SMB Negotiate and Session Setup packets. For each SASL exchange, there is a corresponding SMB Session Setup and Negotiate exchanges.

Upon startup, SASL configures the various plugins it supports. Connected NAS supports the following three plugins:

- GSSAPI/SPNEGO
- NTLMSSP
- Kerberos

All three plugins support the SASL API model for sessions and authentication steps.

GSSAPI/SPNEGO

GSSAPI is considered a wrapper protocol. The data received from the negotiation and session setup exchanges are considered just blobs of data from an SMB perspective. Without a wrapper of some sort, there would be no identifying characteristic of the data. GSSAPI wraps the enclosed data with a small envelope and identifies that data. The enclosed data, in our case, is an SPNEGO packet.

SPNEGO is a negotiation protocol. It provides a simple protocol that allows a server to present various supported security protocols that can be scanned by the client. The client will then be able to select the desired protocol. For ConnectedNAS™, if a server supports Kerberos, and if the client has been built with Kerberos support, we will select Kerberos as the method. Otherwise, we will select NTLMSSP if presented. If, by chance, the server supports neither Kerberos nor NTLMSSP, the client will refuse to authenticate.

In addition to negotiating the authentication protocol, GSSAPI with SPNEGO also wraps the specific authentication handshake. Depending on the authentication mechanism, the GSSAPI/SPNEGO layer will configure the session with a child SASL layer as Kerberos or NTLMSSP. As the authentication handshake progresses, GSSAPI/SPNEGO will unwrap the presented authentication data, pass it off to the child SASL layer, and then wrap the output from the child with GSSAPI/SPNEGO and return it through the SASL infrastructure to the SMB application.

The GSSAPI/SPNEGO layer also provides a conduit for obtaining the negotiated session key from the underlying mechanism.

NTLMSSP

NTLMSSP is the authentication mechanism used for workgroup or server authentication. This is a type of authentication that involves just the SMB client and SMB server. The server creates what is called a nonce, essentially a random number, and sends this to the

client. The client creates an md5 hash with this nonce and the password (and other information) and sends the hash back to the server. Note that the password was never sent in cleartext. The only thing sent was the md5 and there is no direct way of reversing the md5 to derive the password.

The drawback of NTLM is that given enough time, a hacker could simply attempt to regenerate the hash given the nonce and all possible passwords. This is why password length and construction are critical. For example, a simple 5-character password can be cracked in about a second. A 12-character password with a combination of letters and characters can take millions of years.

This password hash is used to generate a session key which is shared between the client and the server, yet never sent over the network. The session key, in turn, is used to both sign and/or encrypt the SMB traffic.

ConnectedNAS™ has implemented all the crypto algorithms required for generating and verifying this hash so no third-party packages are required.

Kerberos

Kerberos is the authentication mechanism used today by most enterprise file servers. It is the basis of Active Directory and Microsoft® Domain Controllers. With Kerberos, A client authenticates with the domain controller and obtains a "ticket" as a result. This ticket signifies that the client has successfully authenticated and can be subsequently used to authenticate with multiple network services. The domain controller may physically be a different computer than the computers hosting the network services.

Separating the authentication server from the target service allows the reuse of the ticket for multiple services on multiple servers. This is referred to as "single-signon". A related ability is to reuse the ticket across multiple clients on the same computer. After one client service has authenticated and obtained a ticket, the ticket can be reused by other client services. This utilizes the notion of a "credential cache" in Kerberos terminology. Tickets are granted with a defined lifetime and will expire. Tickets may need to be renewed if they continue to be needed. Typically, the first ticket in a credential cache is the default or login ticket.

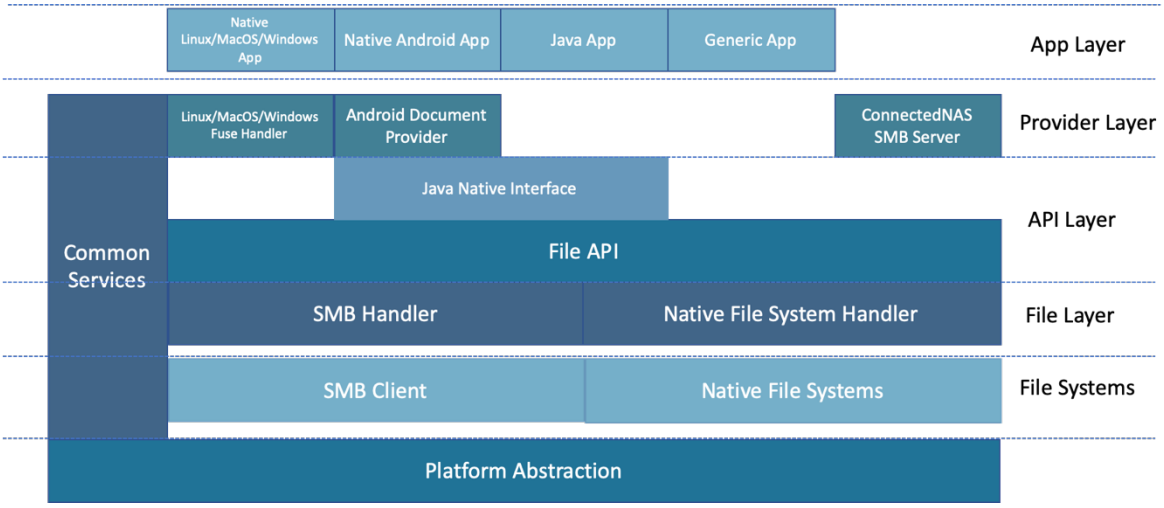
In a simple client/server authentication model, a client authenticates with a server using a username and a password. With Kerberos, a client authenticates using a principal and a password. A principal is similar to a username that is augmented by a service name and a "Fully Qualified Domain Name" (FQDN) using the form service/user@fqdn. The service for SMB has been defined as "cifs". So, an SMB principal for a user named "scientist" in the domain "kdc.lab.com" is **cifs/scientist@kdc.lab.com**.

In the event that a network client does not support Kerberos, it may still be able to authenticate with the domain by using the target server as a Kerberos Proxy over NTLM for the client. That is, the client can provide the credentials through NTLM and the server can authenticate with the directory server on behalf of the client.

A Kerberos implementation would be a significant undertaking to write from scratch. There are two leading Kerberos implementations: MIT and Heimdal. ConnectedNAS™ deploys the MIT Kerberos package with the SASL layer. The MIT Kerberos package is relatively portable and can be made available on most all modern RTOSes. If you find you need a Kerberos stack on your platform feel free to discuss the porting effort with ConnectedWay.

ConnectedNAS™ SMB Client Architecture

The overall architecture of the ConnectedNAS™ SMB Software Stack is shown in the following illustration.



Some description of the layers relevant to authentication and their function follows:

Application Layer

The application layer is the application that issues the file I/O requests. This is typically third party or customer code that wishes to be enabled with access to a network file system. For example, the application layer could consist of code like Microsoft Word®, Oracle®, a Video Player or any other piece of software. There are five basic types of Application models supported by ConnectedNAS™:

- A Native Linux, Windows®, or MacOS® application using a FUSE framework.
- A Native Android® application using the Android Storage Access Framework
- A Native iOS® application using the iOS Document Provider Framework
- A Java application using the ConnectedNAS™ java.io JNI layer
- A generic application using one of the ConnectedNAS™ APIs directly.

There are two types of application layers, those that use a provider layer and those that do not. If the application layer does not use a provider layer, it should handle error returns that indicate authentication failures, query for updated credentials, and reissue the request. This will be discussed more in the section on use cases (ConnectedNAS™ Authentication Use Case).

Provider Layer

This layer is relevant in select OS architectures where access to storage can be provided by a storage provider. Different OS's support different storage providers:

- Android® provides the Storage Access Framework that can provide Document Providers. ConnectedNAS™ for Android includes a Document Provider
- iPhone® OS provides the iOS® Document Provider framework.
- MacOS® provides a User Mode File System driver called osxfuse (osxfuse.github.io).
- Linux provides a native User Mode File System driver called fuse

- Windows® provides a User Mode File System driver called dokan (dokan-dev.github.io).
- ConnectedWay™ provides an SMB Server storage provider that turns any enabled ConnectedNAS™ device into an SMB Server.

The provider layer, if used, will handle authentication failures of requests. Refer to the section on Use Cases (ConnectedNAS™ Authentication Use Case) for more detail.

API Layer

Below the Provider Layer is an API Layer which provides various interfaces to applications that wish to interact with the SMB client. ConnectedNAS™ provides the following APIs:

- WinAPI. The native ConnectedNAS™ File API. Most Windows File APIs like CreateFile, DeleteFile, ReadFile, etc. are supported including Overlapped I/O, File Attributes, and Handles.
- Posix. For ease of porting existing Posix applications, ConnectedNAS™ provides a Posix API that interfaces to our SMB client.
- Java I/O. ConnectedNAS™ provides a Java Native Interface (JNI) that extends existing virtual machines with SMB Client support. The JNI is modeled after the java.io framework.
- RTOS APIs. For select RTOS's, ConnectedNAS™ provides a RTOS compatible API set. This allows for easy porting of existing RTOS applications.

All ConnectedNAS™ APIs support an extension to file naming compatible with the APIs naming scheme. This extension presents a flat namespace for network and native files.

File Layer

The File layer provides file I/O redirection to various file system handlers. These file system handlers include:

- The SMB Client
- A Darwin (MacOS) file API
- A Posix API compatible with Linux and Android.
- A native Windows and Windows Mobile API
- A Virtual File System
- A Pipe File System
- A Mailslot File System
- ThreadX's FileX and Mentor Graphics NuFile file systems.

SMB Client

The SMB Client is the SMB v1/v2/v3 protocol engine and connection session manager. The details of this layer are beyond the scope of this paper, but it is important to note that the SASL authentication occurs within this component. So, an authentication failure will be detected in this layer and passed up to the higher layers.

It's also worth noting that the SMB client is a "real-time" component. That is, the component is non-blocking, and event driven. Events are either application requests, network packets received or timeouts. The SMB client can handle multiple client sessions to multiple servers. It is also single threaded although multiple clients can be run on SMP systems.

ConnectedNAS™ File Naming

A common component of all the various APIs and the underlying SMB client is the file naming. In order to provide a flat namespace for files, we have encoded credentials, servers, shares, paths, and file names all in one global path supported by all components of ConnectedNAS™. This encoding is referred to as the "Universal Naming Convention".

It is of the form:

[\\[username[:password[:domain]]@]server[:port]\share]\path\file

Either a forward slash (/) or backwards slash (\) can be used interchangeable in the name. The '%' character is an escape character and can either escape another % or a two-digit hex number representing the hex value of an ascii character.

All of the following are valid names:

Name	Description
/home/user/file.name	A Posix style path to file.name
//server/share/path/file.name	A file in the share 'share' on server 'server' with relative path 'path/file.name'. The server is accessible at the default SMB port of 445. This will use default login credentials for the server. See section ConnectedNAS™ Authentication Use Case for descriptions on how the default credentials are obtained.
//server:8445/share/path/file.name	Same as previous except the SMB server is accessible at port 8445 rather than the default.
//user:pass@server/share/path/file.name	Authenticate with username 'user' and password 'pass' and access file on server 'server', share 'share', and relative path 'path/file.name'

Name	Description
//user:pass:domain.example.com@server /share/path/file.name	A fully qualified name using credentials for user 'user', password 'pass' in the Kerberos domain 'domain.example.com'

The ConnectedNAS™ File API will return errors to the provider or application layer specific to the API if access to the file or path is denied for security reasons. The provider or application layer can reissue the request with updated credentials.

ConnectedNAS™ Configuration

ConnectedNAS™ SMB Client has two modes of configuration operation: Managed and Unmanaged. In the Unmanaged mode, ConnectedNAS™ will initialize and operate without any persistent state. There will be no saved credentials, or remote shares and the network configuration will operate in a default mode. In Managed mode, the ConnectedNAS™ client will maintain its runtime configuration and restore it after restart.

ConnectedNAS™ maintains its configuration as an XML file. There are two sections of the configuration relevant to authentication: The server section and the credentials section. The server section identifies configured remotes for the client. Within the remote entry, there is a credential field that can be one of three types: a login type, a domain type, or a server type. A domain type refers to an entry in the credential section and is intended to be used for authenticating with the Kerberos subsystem. The server type is credentials specific to that server and intended to authenticate with that server only. A Login type is used to specify that the default login ticket is to be used by the ConnectedNAS™ software. For the login type, the actual credentials are managed outside the context of ConnectedNAS™.

A remote server entry for the ConnectedNAS™ client contains:

- Server name
- Credential type
  - Login Type
  - Domain Type
    - Ticket id (username@fqdn)
  - Server Type
    - Username
    - Domain
    - Password

A credential entry for the ConnectedNAS™ client identifies credentials for Kerberos tickets. The entry contains:

- Principal (username@fqdn)
- Password

Whether the XML file is stored, i.e. whether ConnectedNAS™ operates in Managed mode, is configurable at build time for the target system. Whether passwords are stored is configurable during runtime as a global configuration option. If the XML file is being stored it will be encrypted. The encryption key will be queried during startup of the software package.

See the section on use cases (ConnectedNAS™ SMB Client Architecture) for how this behaves on the various supported platforms.

Credential Manager

Tightly coupled with the configuration manager is the credential manager. There are two interfaces to the credential manager: one for the applications or providers to update and query credentials and one for the authentication component of the SMB client to retrieve credentials.

The credential manager supports functions similar to the Kerberos utilities kinit, kdestroy, kpasswd, and klist. This integration is useful for those platforms with no other interface to the Kerberos subsystem other than ConnectedNAS™.

When the SMB client attempts to authenticate with a remote, it will query the configuration manager using the server name. The configuration manager, in turn, will look up the server in the configuration and determine if the credential is a login, server or domain credential. If it is a login credential, the credential manager is not managing it so empty credentials will be returned. If it is a server credential, then the info in the server configuration will be returned. If it is a domain credential, then info from the credential configuration will be returned.

Failures are returned to the provider or application. The provider or application, in turn, can query and update the credential manager with new credentials and then reissue the request or it can simply reissue the request with a fully qualified universal name (as discussed in section ConnectedNAS™ File Naming).

See the section on use cases (ConnectedNAS™ Authentication Use Case) for how this behaves on the various supported platforms.

ConnectedNAS™ Authentication Use Case

There are two main modes for discussing the authentication use cases: Provider Mode and Direct Mode. ConnectedNAS™ provider mode interfaces include FUSE for Linux, Windows, and MacOS, and the Android ConnectedNAS™ Storage Provider. In provider mode, native file I/O requests pass through the provider layer where the file names are translated into Universal File Names. In direct mode, applications directly interface to the File API layer.

Both the provider model and the direct model support two sub-modes: managed and unmanaged where credentials are managed by a credential manager or not.

The legacy operation of ConnectedNAS™ is direct, unmanaged.

The following use cases will be discussed:

- Applications or Providers using Login Credentials Only
- Application specifying Universal File Names and interfacing with the File API directly.
  - Credentials unmanaged by the ConnectedNAS™ SMB client. NOTE: This is the legacy behavior. (Direct, Unmanaged)
  - Credentials managed by ConnectedNAS™ using the credential manager (Direct, Managed)
- Provider Layer providing mapping of Native File Names to Universal File Names
  - Credentials unmanaged by the ConnectedNAS™ SMB client (Provider, Unmanaged)
  - Credentials managed by ConnectedNAS™ using the credential manager. (Provider, Managed)

Login Credentials

The Login Credentials Use case is really just a special case of the Application or Provider Layer in Unmanaged credential mode. We highlight it separately because it is such an important mode that we see as the primary use case on enterprise deployments.

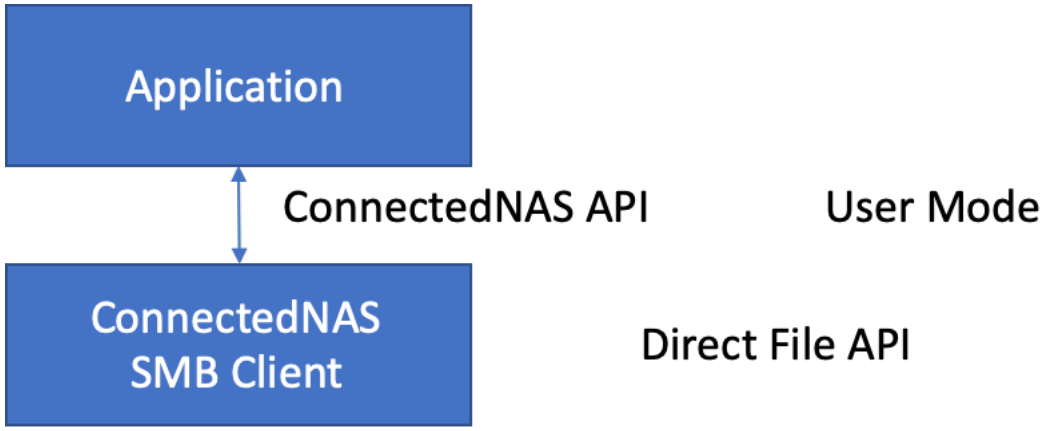
With a login credential, the credential info is maintained outside of ConnectedNAS™ and is why it is similar to the unmanaged case. The login use case only makes sense when using domain login using Kerberos. In the legacy ConnectedNAS™ SMB client using SMBv1, the lack of a credential specification indicated anonymous authentication. The support for anonymous authentication has been deprecated in later versions of SMB due to various vulnerabilities and the fact that a session key, required for packet signing or encryption, is derived from a secret password. Anonymous authentication defeats these security features.

In a domain model, the default authentication ticket is obtained typically upon login. Applications, and Provider Layers do not need to provide credentials to the SMB Client. The client will use the default credential from the Kerberos stack for access to all remote servers.

This can completely free the Application from managing remote credentials.

Direct API

The Direct API is illustrated below:



It's a pretty simple model. The Application interacts directly with the SMB Client using the ConnectedNAS™ API.

Direct API, Unmanaged Credentials

This is the legacy behavior of the ConnectedNAS™ SMB Client. In this mode, the application passes universal file names which include the credential information in the file name, to the File API. This file name is passed into the SMB client, the credentials are parsed out of the name, and used during authentication with the remote server.

If the credentials are missing or invalid, authentication will fail, and an authentication error will be passed up to the application. The application, in turn can query for updated credentials and reissue the request.

For example, see the following code snippet. Comments will be provided italicized preceeding the snippet:

The filename is initialized to some default path, with or without credentials. It is placed in the heap so we can manipulate the file name with credentials as needed.

```
filename = BlueCtstrdup (TSTR("//server/share/path/*");
```

APIs using filenames are wrapped with a while loop so that the APIs can be reissued with updated credentials if needed.

```
while (retry == BLUE_TRUE)
  {
```

Call the File API with the universal file name. NOTE the API is similar to the Windows API.

```
    list_handle = BlueFindFirstFile (filename, &find_data, &more);
```

If we obtained a valid handle, then the API call was successful, and we will exit the loop

```
    if (list_handle != BLUE_INVALID_HANDLE_VALUE)
      retry = BLUE_FALSE ;
    else
      {
```

The call failed. Get the last error and print it for those following along.

```
        last_error = BlueGetLastError () ;
        BlueCprintf ("Failed to list dir %A, Error Code %d\n",
                     filename, last_error) ;
```

Check for authentication failure

```
        /*
         * LOGON_FAILURE can be returned if authentication failed
         * ACCESS_DENIED if authentication succeeded but no access to share
         * INVALID_PASSWORD if password different then connection
         */
        if (last_error == BLUE_ERROR_LOGON_FAILURE ||
            last_error == BLUE_ERROR_ACCESS_DENIED ||
            last_error == BLUE_ERROR_INVALID_PASSWORD)
          {
```

Query and Read updated Credentials

```
            BlueCprintf ("Please Authenticate\n") ;
            BlueCprintf ("Username: ") ;
            BlueReadLine (username, 80) ;
            BlueCprintf ("Domain: ") ;
            BlueReadLine (domain, 80) ;
            BlueCprintf ("Password: ") ;
            BlueReadPassword (password, 80) ;
```

Convert these to Unicode because the API we're using is Unicode. We also support ASCII APIs.

```
            tusername = BlueCcstr2tastr (username) ;
            tdomain = BlueCcstr2tastr (domain) ;
            tpassword = BlueCcstr2tastr (password) ;
```

Update the universal file name with the updated credentials.

```
            /*
             * Now remap the device
             */
            BluePathUpdateCredentials (filename, tusername,
                                       tpassword, tdomain) ;
```

Free up the Unicode copies of the credentials so we do not cause a leak.

```
            BlueHeapFree (tusername) ;
            BlueHeapFree (tpassword) ;
            BlueHeapFree (tdomain) ;
          }
        else
          {
```

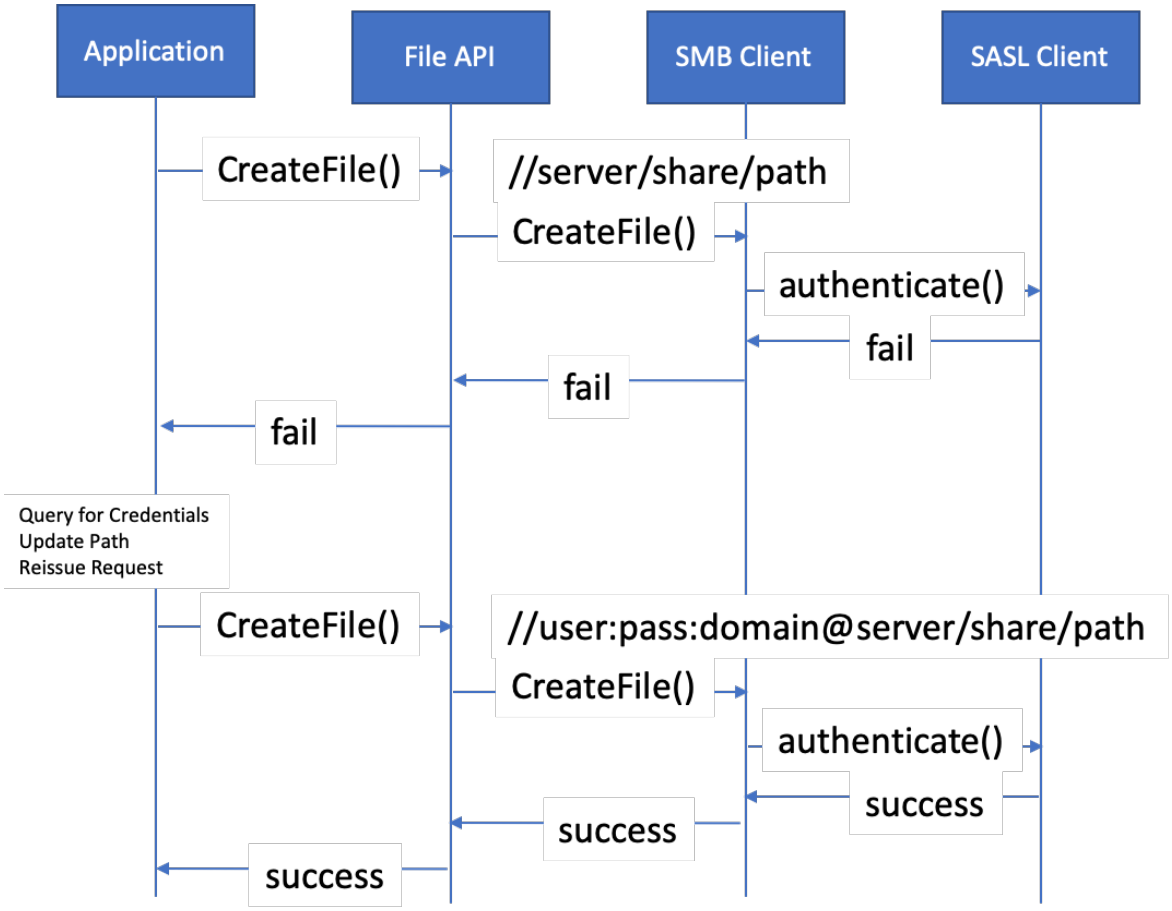
Not an authentication error. Stop retrying and let the normal error recovery handle the scenario.

```
            retry = BLUE_FALSE ;
          }
      }
  }
```

Free the heap copy of the filename. Also, at this point, list_handle will either be a valid handle or the value BLUE_INVALID_HANDLE.

```
BlueHeapFree (filename) ;
```

This sequence is illustrated below:



To use this kind of authentication model, it was recommended that ALL APIs that utilized a file name be wrapped with this kind of wrapper. There are ways this example could be made more generic to minimize code clutter.

The above example is generic for the type of credential: login, domain, or server. If the ConnectedNAS™ SMB Client is being deployed into a domain where all credentials are login credentials, and the login occurs prior to using the API, the example could have been written without the wrapper:

```
list_handle = BlueFindFirstFile (TSTR("//server/share/path/*",
                                 &find_data, &more) ;
```

Even in the absence of a ConnectedNAS™ credential manager, the Kerberos subsystem will use the login ticket by default for all authentication requests. It is clear that through the use of domain credentials, the API model is greatly simplified.

Direct API, Managed Credentials

With managed credentials, the application need not construct universal file names with embedded credentials. Rather, the Application would update the configuration in the credential manager and reissue the request.

To facilitate this, there is a new configuration API added that allows a credential.

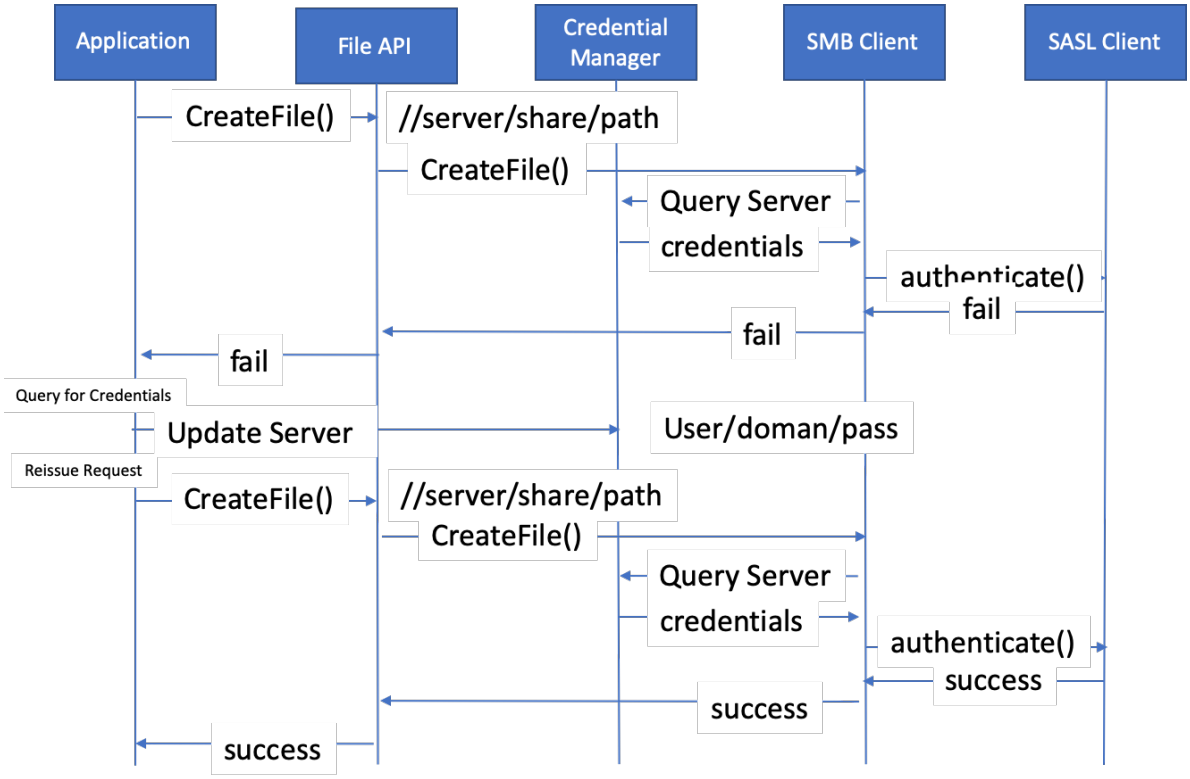
```
typedef enum {CRED_LOGIN, CRED_SERVER, CRED_DOMAIN } BLUE_CRED_TYPE ;
BLUE_UTIL_LIB BLUE_VOID BlueConfigUpdateCredential (BLUE_CHAR *server,
                                                    BLUE_CRED_TYPE type,
                                                    BLUE_CHAR *user,
                                                    BLUE_CHAR *domain,
                                                    BLUE_CHAR *pass);
```

If the server doesn't exist in the configuration, it will be added. The configuration will be updated as otherwise appropriate.

The example would look mostly the same as above. The exception being that the call to BluePathUpdateCredentials will be replaced by a call to BlueConfigUpdateCredential.

The SMB Client, upon receiving a request for a remote session, will call the credential manager to obtain the credential information for the server. The SMB Client knows to query the credential manager based on the absence of credential information in the universal file name. If the server credential is a login type, the SMB client will allow the underlying authentication module (Kerberos or NTLMSSP) to use its default credentials. If the credential is a server type, the credentials will be obtained from the server configuration and if it is a domain type, the credential will be obtained from the credential configuration.

This interaction is shown below:



Instead of calling BlueConfigUpdateCredential within a wrapper as above, it could be initialized before the application is run. This would allow the wrapper to be removed. That is, the application could initialize the managed credentials through a separate user interface that must be invoked prior to accessing files using the API. Authentication failures could simply direct the user to some credential user interface managed by the application.

The advantage of using the managed credential model is that the credentials are remembered. If the configuration file is stored and if passwords are remembered, then once the credentials are updated, there would be no need to provide credentials in the API calls and no need to check for authentication failures other than to report them. Whether

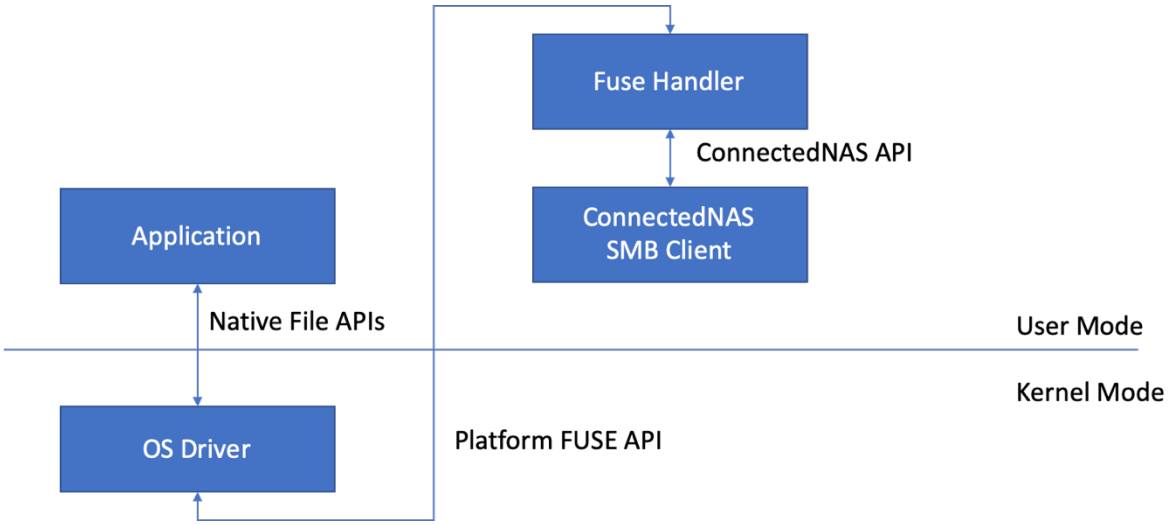
credentials are updated within a wrapper, or whether they are updated as part of a credential user interface is determined by the application developer.

Provider Model

A provider layer is used to translate native file requests to network aware universal file request and to interact with the ConnectedNAS™ File API. As mentioned elsewhere, ConnectedNAS™ provides Linux, Windows, and MacOS FUSE layers, and Android and iOS Document Providers as provider layers.

When using a provider layer, the responsibility for providing credentials and for managing those credentials falls within the provider layer.

This is illustrated below:



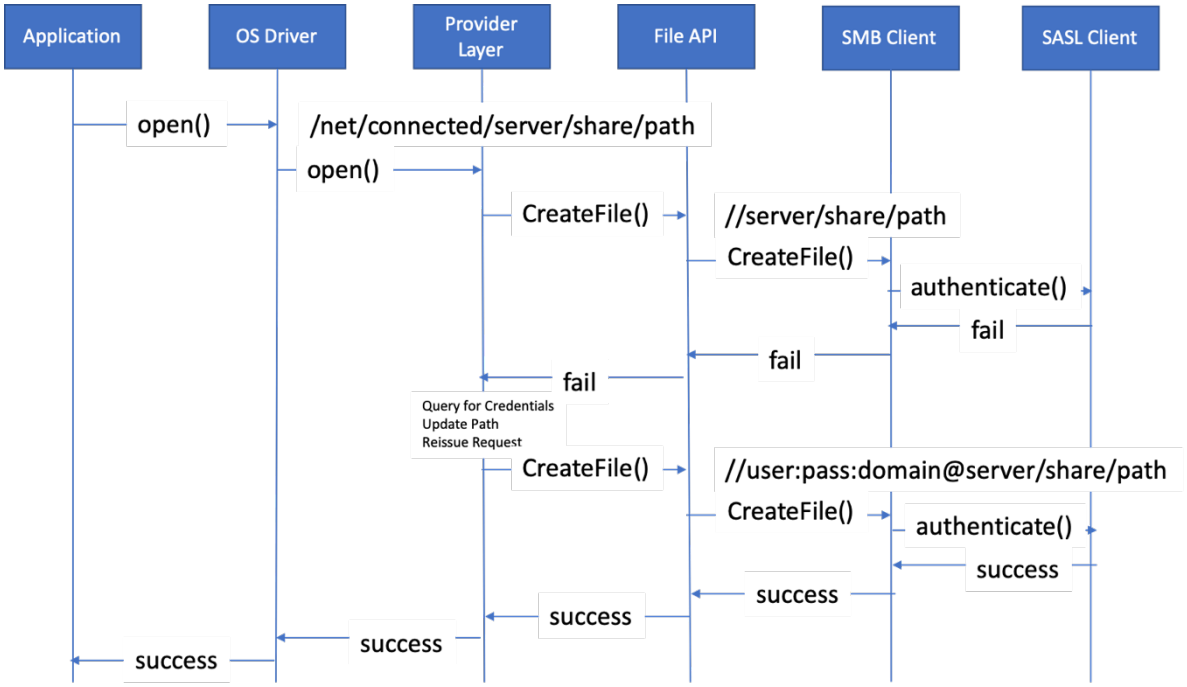
Provider Layer, Unmanaged Credentials

Both FUSE and "document provider" provider layers use a map to translate between the native file name and the universal file name used by the ConnectedNAS™ SMB Client. When the provider layers are using unmanaged credentials, it is this map that is updated with the credentials.

This is the way our initial version of the Fuse and Android document providers work. The remote servers and their respective maps are initialized with credentials that were passed into the SMB Client as part of the universal file name.

Upon authentication failures, the provider layer could query the user interface for updated credentials and update its map. This would imply that the provider layer has an authentication loop similar to that used in the Direct API case. In fact, the discussion of the Direct API, Unmanaged Credentials use case applies to the provider layer case as well. The difference is that the interaction for the credentials is performed in the provider layer and is completely hidden from the application.

This sequence is illustrated below:

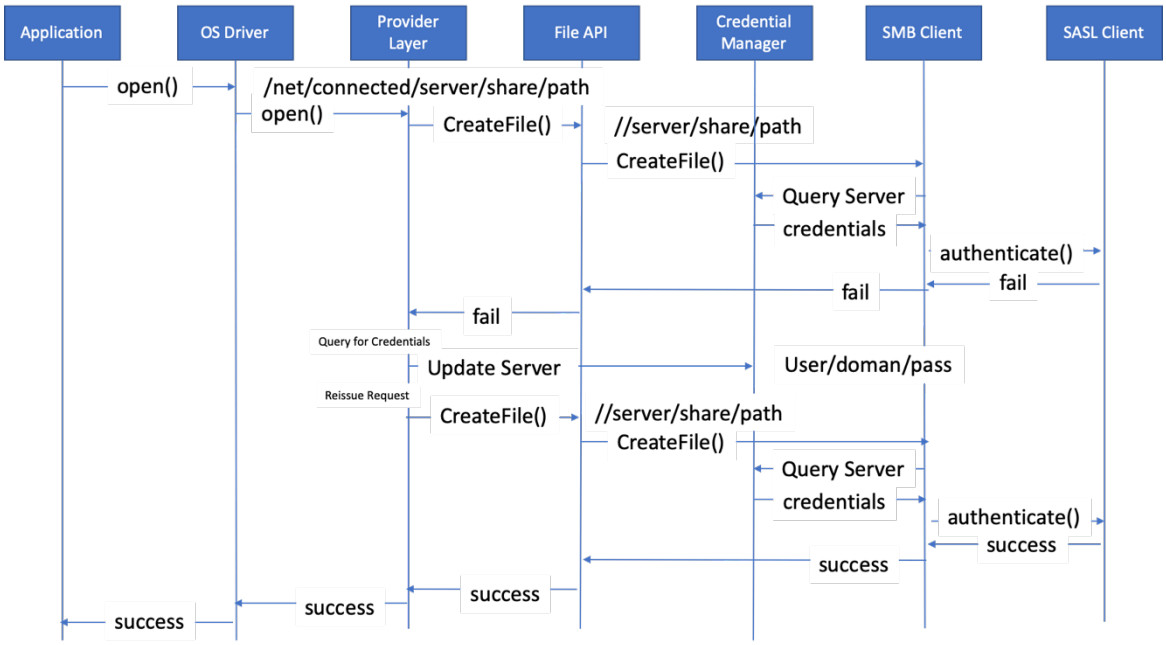


Provider Layer, Managed Credentials

With managed credentials, the provider layer need not construct universal file names with embedded credentials and its map need not include any credential information. Rather, the provider layer, upon authentication failures, updates the configuration in the credential manager and reissues the request.

The discussion in the section, Direct API, Managed Credentials is also relevant to this mode with the exception that the authentication error handling occurs within the provider layer rather than the application layer.

The Sequence Diagram is shown below:



Conclusion

This white paper has outlined the basic architecture of the ConnectedNAS™ SMB Client from the perspective of authentication and privacy. It has also introduced the packets used by the SMB protocol and the security related fields within those packets. Lastly, this paper discussed the Use Cases supported by the ConnectedNAS™ SMB Client that can leverage both the architecture and the protocols for offering the most robust and secure remote file access capabilities available.

It is hoped that after reading this white paper, the reader will have a good vision of how they could deploy ConnectedNAS™ for their unique application requirements. If, after reading this paper, there are still open issues in how ConnectedNAS™ can be deployed, please do not hesitate to contact the author.