
Saxonite Limited

ArtiSan Product Information

A Short Overview of NT Security

Author: Keith Young
Reference: ArtiSan Security
Date: 25 June 2005
Version: 1.0
Distribution: Company confidential

Abstract

This document is a brief overview of security in W2K and W2K3 with specific reference to the ArtiSan product.

© Saxonite Ltd 2005

Change History

1.0 Initial

Contents

Introduction	2
The NT Security System	4
Access Rights, Resources and Principals	4
Privileges and Policy	9
COM Security	10
Local and Domain Security	11
Rights and Privileges in Operation	13
ArtiSan Rights and Privileges	17
The ArtiSan Install Process	17
Internet Explorer Permissions	20

Introduction

This document is a brief description of the main concepts that underpin the W2K and W2K3 security system. It is designed to act as an introduction for people installing and supporting ArtiSan. It is often easier to make changes to a system if the reasons for those changes are clearly defined and presented within a well-understood framework. This document attempts to present one such framework.

W2K and W2K3 security systems are very similar to each other; the differences usually relate to what is default behaviour. Where we are highlighting a particular difference between them we will refer to the specific system. When we want to refer to the general case, we refer to the operating system as NT.

Security underlies many of the activities of the NT system and is a major consideration in the behaviour of many OS and user-level processes. In NT, security is central to the OS architecture having as important a role as memory management, process management or threading in the kernel. However, the kernel relies on external system processes to retrieve information concerning the context in which the security information is applied. For instance, the login prompt on an NT system (the process is known as GINA or Graphical Identification and Authentication) is an application-level process that acquires and authenticates user credentials and constructs a token associated with the user. (Don't worry if these words are unfamiliar, they will be explained later). The point to make is that the OS relies on GINA and similar processes to create the user context in which it can evaluate the lower level access rights. The application programmer also has a responsibility to provide the appropriate security information when using the various OS interfaces.

ArtiSan relies on the deployment engineer to install and configure a variety of system components in order to create a security context in which the ArtiSan software can operate. This is necessary because unlike some other application software, ArtiSan is tightly integrated into the domain in which it resides requiring an appropriate security context in which to access resources such as Exchange and ADS.

In the first part of the document, we will deal with some of the concepts that underpin NT security. I have tried to make this document as accessible as possible for people who know very little about NT security. Inevitably, that means that, for people who know more, much of the information covers "old ground". Also, as a result, this document is a simplification of what is actually happening in practice, but the model presented works for all but the most complex cases. The important thing is to achieve a basic framework or conceptual model for the security system as a whole, because it is only with such a model that faults can be diagnosed and resolved.

One of the problems with NT security is that there is a specialised terminology associated with it. This terminology is useful because it is very precise, but it suffers from being unfamiliar to many people. I will try to identify specialised terms I have used in the context of the text by highlighting them in red.

The second part of the document is intended to take the information presented in the first part along with the current install guide and attempts to:

- a) Explain the purpose of the installation step
- b) Indicate the likely effects of not performing the step

In some cases, there are tests that can be performed that indicate whether the action was successful. In this way, this document is also meant to act as additional guidance for the install process in V1.1 of ArtiSan. In V2 and above, many of the install steps are automated, but the process being undertaken is essentially the same.

In some senses, this document is both an application note and a trouble-shooting guide, but the format of neither of those seemed appropriate so I have presented the information in the form given here in the hope that this will suffice. It should be remembered that this document is a living entity. As the security requirements of the ArtiSan software evolve, so should the document. In that sense, the document can never be complete.

Further Reading

There are a number of standard texts available that describe the NT security system from different perspectives or for different purposes. For a high level overview of how the kernel implements security, see Helen Custer's "Windows NT". For a more technical perspective of how the kernel implements its security features, see "Inside Windows 2000" by Solomon & Russinovitch. For developers, the most important book is Keith Brown's "Programming Windows Security". Keith Brown covers just about everything from basic user account through to COM security. There are also many resources on the web including TechNet and MSDN.

The NT Security System

The NT security system depends on the interaction of various components at the application level and a large chunk of software called the Object Manager that is located in the NT kernel. System service providers, such as LSASS and the SSAPs, authenticate users and acquire privileges on their behalf. Interactions between application level code and the kernel are passed via the Object Manager to validate the rights of the user to perform operations on the various resources they are using. Administrators control security through the granting of access rights and privileges to user accounts. Rights and privileges may be granted either locally or at the domain level.

Access Rights, Resources and Principals

Key Players

The key players in the security system are the **Principal**, the **Resource** and **Access Rights**. At a simple level, a principal such as a user or a group is given access rights to a resource. For instance, a user such as the Administrator can be given Read access rights to a resource such as a file. When the OS attempts to open the file, on behalf of a user it checks whether the current user has sufficient rights and grants or denies access accordingly. Windows documentation uses the terms principal and **trustee** interchangeably.

The possible access rights associated with a given resource depend on the **Resource Type**. This is obvious when you come to think about it, because the kinds of thing you can do with a resource change depending on what the resource is. For instance, a directory has an access right to "List Children". Such a right would make no sense for a file, as the file has no children. Generally, there are two types of access right, the generic ones that apply to all resources and the specific ones that only apply to a specific resource type.

Generic rights include rights associated with using the resource such as read, write and delete rights and specialised rights associated with the security system. These latter rights allow or deny principals the right to change the security rights of the resource.

In NT, the access rights associated with a principal are contained in an object called an **ACE** (Access Control Entry). Each ACE specifies the access rights associated with a given principal. So to specify access rights for two principals, you need to create two ACEs. ACEs are associated with a resource using a list called an **ACL** (Access Control List). Each resource has an ACL. The ACL for a resource can contain many ACEs for a given principal, but each ACE refers to only one principal at a time. Remember that a principal can be a group, a user or some other entity. In reality, every resource actually has two ACLs known as the SACL (System ACL) and the DACL (discretionary ACL). The SACL is used for system auditing whereas the DACL is used to control access when using the resource. We are only really concerned with the DACL, so when we refer in the rest of this document to the ACL, we mean the DACL.

In addition to the ACL, each resource also has an owner and primary group. Owners are not really all that significant in ArtiSan. Owners implicitly have write access to the ACL. The main effect, visible to end-users, of having an owner is that the owner of a resource can always get access to it irrespective of the ACL. Suppose you had a file with an empty ACL. This would imply that nobody could do anything with it, including setting up the ACL to give people access to it. The file would become useless. However, the system will always allow the owner to add ACLs to the object to make it accessible. The primary group of a resource is used in the Posix sub-system and has no particular impact in Windows. The combination of all the security information associated with a resource is collected together in a structure called a **Security Descriptor**.

To see an example of the above, right click on a resource such as the "Program Files" directory on a computer and select "Properties". Now select the "Security" tab. In the top list box you will see the list of principals that have an ACE associated with them, in other words the ACL. In the bottom box you will see the rights associated with each principal.

Some are generic and others are specific to the resource. This display is, actually, a summary. The real ACL is displayed by clicking on the “Advanced” tab where you can also see the Owner. XP also provides a display that shows the effective permissions taking into account local and domain settings.

There are several predefined principals that can be placed in ACEs whose value is determined at the point of access. Examples of these include:

OWNER – This represents the user account that currently owns the resource. Principals can have the right to “take ownership” of a resource. The consequence is that as the owner changes, the rights that were associated with this ACE are transferred to the new owner.

Authenticated Users – This represents any user that has authenticated with the system

EVERYONE – This represents any user of the system. This is a commonly used to allow unrestricted access to a resource such as a directory.

SYSTEM – This represents a special account used by the kernel and some user level system processes. Most notably, this is the default account for NT Services.

There are also specialised well-known principals that represent the kind of login context the user has (e.g. terminal services client, a service, an interactive user) or principals that refer to the role of a computer in the network (Domain Controller, Domain Member).

Thus, each ACE contains a principal and a set of access rights. It also contains some bits that describe how the ACE should be applied. One of these bits stores the type of ACE. An ACE can either be an Allow ACE or a Deny ACE. For instance, suppose there is an ACE that has the principal as the Administrator, the access rights set to Read and it is an Allow ACE, this means that the Administrator is allowed to read the resource. Equally, if the resource has an ACE that is a Deny ACE with the same principal and rights, then this means that the local Administrator is not allowed to read the resource. Deny ACEs always take precedence over Allow ACEs, so if a resource has both these ACEs in its ACL, the result is that the Administrator is NOT allowed to read the resource because the Deny ACE overrides the Allow ACE. The order of the ACEs in the ACL has no significance.

It is very easy to “play” with all this by simply creating a directory with a file or two and simply changing the rights associated with the directory and files and see the effect.

What are Principals?

Principals or trustees can be users, groups, computers or other entities. When a computer boots up in a domain, the computer actually logs into the domain in much the same way as a user would log in. You can see this if you enable the audit logs to audit successful logins and boot the machine. A computer login is display as the name of the computer followed by a \$ sign. It is important to understand that all operations in NT (even booting) have to be performed within the context of a principal. However, for the most part we will be dealing with principals as either users or groups.

Groups are interesting in NT because they are simply containers for other principals. Consequently, a group can be added to another (under some circumstances). This allows the system to set up default access rights in a way that requires very little user intervention. For instance, when you add a computer to a domain, the Domain Admins group is automatically added to the local Administrators group so that domain administrators can control the member computer.

Some principals are user-defined and others are built in to the system. When you open the “Active Directory Users and Groups” console on a domain controller, you can see the distinction between a built-in account and a defined account in the tree view on the left. For instance, the SYSTEM account (also known as “Local System” and “NT AUTHORITY\System”) is a built-in account on all machines (though it is not visible on a domain controller for reasons given below). However, the SYSTEM account on one machine is a different account from the SYSTEM account on another. Alternatively, the Administrator account on a machine is defined at the time the computer OS is installed (you do this when you provide the machine the Administrator password during the OS

install process). The Administrator account is, actually, user-defined account, not a built-in one.

As an interesting aside, the Administrator does not have intrinsic rights built into the OS as for instance does the root or super-user account in Unix. Administrators can only administrate by virtue of having the appropriate rights in the ACLs. Consequently, it is common to allow Full Control of a resource to the appropriate administrative accounts simply to allow administrators to control the system. On a local machine, any member of the Administrators group is regarded as an administrator. However, this imposes some restrictions on how an administrator should behave. In general, all employees (including administrators) should perform their normal daily activities, such as communicating via email, writing reports, etc. in a non-administrative context. When an administrator wishes to act as an administrator of a system, they should log in to an administrative account. This is a very basic “Best Practice”.

In a domain there are many kinds of administrator depending on what resource is being controlled. For instance, there are:

Enterprise Admins – people who administer collections of domain controllers

Domain Admins – people who administer a domain

Exchange Admins – people who administer Exchange

Sometimes, there are strict business and compliance reasons for making these distinctions. For instance, Domain Admins actually have very restrictive rights when it comes to Exchange content. In general, NEVER suggest making any changes to domain level administrative groups, a refusal often offends. ArtiSan does not require any changes to domain level administrative groups.

SIDs and SAM Account Names

Principals are given a short name that is used to identify them during the logon process. This is sometimes known as the **SAM Account Name** or “login name”. For example, Administrators, Operators and fblogs are all examples of SAM account names.

The OS on the other hand uses a unique identifier known as a **SID** (Security Identifier). A SID is a binary structure that uniquely identifies a user. Most commonly, you see a SID represented as a string that looks like “S-1-5-23...” These are called **SID Strings**. The SID is not only a unique identifier, but also carries with it the chain of authorities that authenticate the user. A local machine SID tends to be short because the chain of authorities is simply the local security database. A domain SID is longer because the chain of authorities also includes the domain controller.

Generally, the operating system never records SAM Account names, but instead looks them up on the fly. This is why you are allowed to change the SAM Account Name for a user without breaking anything. You cannot, however, change the SID associated with an account. Consequently, ACEs store the SID, not the SAM Account Name. If you delete a user from the system, it does not delete all the ACEs associated with that user, but it does delete the SAM Account Name from the security database. If you display the permissions on a file, the rights for any deleted users show up as SID Strings because the system cannot find the associated SAM Account Names anymore. Incidentally, the same is true of email addresses. This is one reason why it is conventional to never delete users, but rather simply disable them. A SID is always unique in a system even if you delete the user associated with it. The system never reuses a SID. There are tools to tidy up systems having ACEs associated with deleted users in the Windows Resource Kit.

There are two types of SID, well known SIDs and user SIDs. Well-known SIDs are assigned to specific principals that are known to the system. For instance, well-known accounts and roles such as Administrator, Domain Admins and Domain Users all have well-known SID values. If you were to logon to a French computer, you would find that the administrator account is called “Administrateur” or similar. However, the SID for the administrator account is the same as for a US install. A user SID is constructed when you create the principal.

Authentication

Authentication is the process by which a user validates that their credentials are correct. By **Credentials** we mean the pieces of information that identify the user, i.e. the SAM Account Name, the security database for which the user name is valid and the password.

Authentication happens by one of two mechanisms, known as “pre-authentication” and “Challenge and Response”. In pre-authentication, the user supplies the credentials as part of the request. In Challenge and Response, the user is challenged at the point of request and must provide the credentials in reply to the challenge. Which process is being used depends on the resource being accessed.

The system validates a set of credentials using a Security Service Provider (they are sometimes called SSPIs because the API they implement is called the Security Service Provider Interface). The provider used depends on:

- a) What is available (different versions of NT support different SSPIs)
- b) What is configured (SSPIs can be added on the fly)
- c) What is default (generally, most applications do not specify an SSPI but simply use the default)

The task of the SSPI is to take the credentials and authenticate them with the security database. Different SSPIs use different strategies to do this and the strategy used determines to some extent what rights can be granted to the user.

In Windows 2000 and above the default security provider is Kerberos whereas in NT it is NTLM (NT LAN Manager). Windows 2000 supports both providers so that you can maintain networks with a mixture of NT and W2K boxes. W2K also supports other providers that are used in specific circumstances such as where a non-Microsoft security database is being used. Actually, additional security providers can be added to the system. In some very secure environments this is how the additional levels of security are achieved. Security providers vary in the features that they provide. NTLM is actually a very limited security provider, Kerberos a much more powerful one.

In some circumstances, NT will accept credentials without specifying a security database (i.e. not require a domain). In actuality, it simply automatically assumes that the user name and password are valid for the default security database for the machine. If the machine is a member of a domain name, it uses the domain and, if not, it uses the local security database. However, it is not wise to assume this behaviour because it does not always work with all software.

What are Resources?

In NT, resources can be almost anything. The obvious ones are Files and Directories, but resources also include:

- a) Hardware devices such as disks, tapes, printers, USB ports
- b) Logical devices such as disk volumes, multimedia devices, network pipes
- c) Registry keys and values
- d) Network shares and other network providers such as the RPC server and named pipes
- e) Mailboxes and mailbox folders

Each resource has a built-in mechanism that allows it to store the ACLs associated with the resource. For instance, in NT, files and directories store permissions in extended properties associated with them. This means that as you define permissions associated with a file or a directory, the permissions use up additional disk space. File ACLs have a maximum length that is constrained by the maximum size of the extended properties. In NTFS, extended properties can be no more than 64K bytes in length. The maximum number of ACEs in the ACL for a file depends on the length of the SID in the ACEs, but is typically about 1800 or so.

Hardware devices such as disks and tapes store the permissions in parts of the registry associated with the device driver used to access the device. Permissions to mail objects are stored in the Exchange Stores and, at a higher level, in the ADS.

Network shares are interesting because they have two levels of permission. There are permissions associated with the share and permissions associated with the underlying file system being shared. Suppose you have a file system exposed by a SAN that does not provide NT-style security. This is typical as most SANs use a protocol called SMB (Server Message Block) to expose shared drives and printers. SMB is the protocol used by old 16-bit windows boxes (and DOS) and has no intrinsic knowledge of permissions on a file-by-file basis. SMB servers only allow permissions to be set at the share level not the file level. NOTE: Prior to version 3, ArtiSan did NOT support this type of SAN. The file share software provides a set of permissions associated with the share itself. This allows administrators to control access to the SAN. These permissions not only affect who can access the files on the SAN but also whether the share itself is visible to an end-user. You can see this in NT by right-clicking a directory and selecting the Sharing tab. The permissions button allows you to specify who can access the share (and whether it is visible).

Now, suppose the shared directory was on an NTFS partition. Then there would not only be share permissions, there would also be the underlying file permissions (viewed on the Security tab). For a client to access a shared NTFS drive, they must have the right access rights to perform an operation on BOTH the share permissions and the file permissions. The most common case where people come across this issue is when dealing with Web and FTP servers. Typically, you set the access permissions at both the server level and the file level. ArtiSan uses this facility when the underlying archive is a remote drive.

Access Rights Inheritance

We have noted that the ACLs for a resource need to be stored somewhere. Suppose we had to store all ACLs for all resources on a computer. We would end up using an enormous amount of storage space just to store the ACLs. Moreover, any change to the permissions would require that we have to redefine those permissions at all levels, which would be an administrative nightmare. To get round this problem NT has a notion of inheritance of permissions.

In NT, many resources are “contained” in some kind of parent resource. For instance, a file is contained in a directory, which in turn is contained in another directory. The root directory is contained in a logical volume and so on. Each resource has some bits that determine whether the permissions on the resource can be propagated from the parent resource and whether its permissions propagate to its children. The same can be said of a mailbox folder. The folder has a parent folder that is, in turn, contained by a mailbox that is contained by a mailbox store that is contained by an Exchange server. For all such relationships there is a top level or root object from which all permissions may ultimately be derived.

Typically, the end-user defines the permissions at some point in the hierarchy and allows those permissions to be inherited. This means that:

- a) The user only needs to change permissions at one level to affect large numbers of resources
- b) The ACLs only need to be stored at the top level

Now, suppose we want to further restrict access to a specific file or directory. To achieve we could simply add an extra ACL onto the relevant resource. Alternatively, we could decide to remove the “inherit from parent” bit from the resource and add the specific permissions for the user or users we are interested in. The operation we perform depends on what we are trying to achieve. ArtiSan does this to restrict end-user access to the underlying files in the archive.

In addition to access rights, administrators also grant users system **privilege**. A privilege is the right of an account, such as a user or group account, to perform various system-related operations on the local computer, such as shutting down the system, loading device drivers, or changing the system time. Privileges differ from access rights in two ways:

- Privileges control access to system resources and system-related tasks, whereas access rights control access to securable objects.
- A system administrator assigns privileges to user and group accounts, whereas the system grants or denies access to a securable object based on the access rights granted in the ACEs in the object's DACL.

There is a key pair of privileges called “Act as part of the operating system” (a.k.a. `seTcbName` or `SE_TCB_NAME`) and “Impersonate a client after authentication” (a.k.a. `seImpersonate` or `SE_IMPERSONATE_NAME`). These privileges allow a user to log on as another user and to impersonate them (i.e. they are very powerful privileges). There are other privileges that allow you to control whether a given user can logon to a machine and, if so, what kind of logon they can do. For instance, “Logon Locally” allows you access to a desktop, whereas “Logon as a Service” allows you to specify the account as the login for a given NT Service. Other privileges allow users to shutdown a computer or to install or remove hardware devices and so on. There are about 20 basic privileges, but they change with different versions of NT.

For ArtiSan to function, it requires that some additional privileges be granted to the domain ArtiSan user and the local ASPNET or NETWORK_SERVICE user.

Privileges are controlled through **policies**, which are simply collections of privileges and other information. Policies can be applied either locally or at the domain, tree or forest level. Policies can also be defined separately for different organisational units within a domain. Each can add to or override the previous. The other key difference is that privileges have to be both accessible and enabled. However, generally, the application programmer hides this last distinction.

On ordinary NT hosts that are not in a domain, policies are determined by the Local Security Policy application, a system tool accessible through the Control Panel. However, on domain members, policies are defined both in the same tool and using Group Policy Objects (GPOs) defined at the domain controller. The domain controller has a “Domain Controller Policy” and “Domain Policy” editors that are simply summaries of the content of the GPOs in force.

To simplify this process, administrators are supplied with a set predefined policy templates. For instance, there are policy templates with names such as:

Hisecdc.inf – a set of policies for a high security domain controller

Basicsv.inf – a basic policy for a member server

Securewk.inf – a policy for a moderately secure workstation

For the most part, domain policies are used to control what a user on a machine can do because it allows a domain administrator to centralise the control of their network.

In reality, this is a simplistic view because GPOs are not only used to control privileges, but also can be used for all sorts of other administrative tasks such as software distribution, security auditing, remotely setting application configurations and so on. For instance, a common use of GPOs is to automatically configure MSIE security settings on user desktops. This is useful in ArtiSan because it allows the domain administrator to define the ArtiSan server as a member of the local intranet within the domain. The consequence of this is that end-users are unlikely to need to constantly re-authenticate when accessing the ArtiSan server to view items. Policies can also be used to set specific registry variables or copy files to specific locations.

One problem with GPOs is that they take time to propagate from the domain controller to the domain member. Changing a setting may take some time to be reflected on a member

server. The key to this is to look at the “Effective Settings” for privileges in the Local Security Policy Editor. There is also an MMC add-in called the RSoP (the Resultant Set of Policy) that allows you to view the current effective policy as applied to a particular user on a particular machine.

COM Security

At this stage, we must digress to another area of security that is no less important in the ArtiSan installation, COM security. ArtiSan uses COM at its most fundamental levels. All ArtiSan interfaces are based on COM. The ArtiSan Requestor object is a COM service; the RequestProxy is a COM server.

As stated above, security can be implemented by network services at arbitrate access to network resources. Our example earlier of file sharing software illustrated this. COM extends the notion of a network resource to include COM applications. This makes sense because DCOM applications can be exposed to the network as applications that can be used by other hosts in the network. However, a side effect of this is that access to COM services are also controlled through the same mechanism even if they are only visible to the local box.

COM security is managed through an applet called DCOMCnfg. On W2K3, this has been subsumed into the “Component Services” applet. Either way, the basic interface and the concepts that underpin it are the same. COM maintains a default configuration that determines:

- a) The network protocols that are user between COM clients and servers
- b) The security requirements for launching and accessing COM servers
- c) The communication properties used when communicating between COM clients and servers.

The network protocols in use only affect DCOM (COM communication between machines) and have no impact on ArtiSan.

COM also maintains per service overrides for each of these configurations. For the most part, you should never edit the defaults. But you may need to edit the properties for specific ArtiSan services depending on the customer domain policies. Microsoft has a habit of changing the default values for these permissions each time they release a new version of Windows or even, sometimes, a service pack. You will know that you need to edit these permissions because the event logs will contain messages to the effect that the Requestor service failed to launch or could not be accessed, e.g. something like:

Access denied attempting to launch a DCOM Server. The server is: {8597FF3C-F7BF-4e08-B2E1-23683436D570} The user is XXXXX

Access denied attempting to launch a DCOM Server using DefaultLaunchPermssion. The server is: {8597FF3C-F7BF-4e08-B2E1-23683436D570} The user is XXXXXX

You resolve this problem by granting access and launch permissions to all authenticated users. This is achieved by selecting the application (in the ArtiSan case, the significant application is Requestor service) and view the properties associated with that application.

The Security tab allows you to decide who can access, launch and configure the service. ArtiSan expects any Domain User to be able to launch and access the Requestor service. If the default permissions are more restrictive than this, then you need to use custom permissions on the Requestor object itself. Although the dialogs are a little different from the standard security dialogs, you will notice a lot in common with them. Both are concerned with editing the ACLs associated with the resource (in this case the COM application). The main difference is that each setting is simply editing a right (either access or launch) and so you don’t have to specify the rights required in the dialog. When you edit these objects, you are actually simply editing the permissions of various registry keys and values.

The other tabs are of less interest to us. The General tab deals with the way COM clients should authenticate themselves with the Requestor. You can request that authentication

only happens at connection time (or earlier) or you can specify that the client authenticates on each access to the object. This setting should be left as the default global configuration. The Location tab deals with the location that the COM object should be executed. This should always be set to run the application of the local computer (ArtiSan does not use distributed COM method calls). The Endpoints tab has no significance for ArtiSan and can be safely ignored. The Identity tab indicates the user account that should be used when running the service. This should be automatically set to use the ArtiSan user account.

Local and Domain Security

When the OS is installed, it acts wholly using local access rights and privileges. The computer operates under the control of the local security database (called SAM), authenticating users according to user accounts created by the local administrator. The local administrator defines the privileges associated with the accounts and sets access rights on individual resources.

If the machine is being installed into a domain, then the installer now adds the computer to the domain. Computers can have one of two roles in a domain, acting as either a domain controller or a domain member. If the machine is intended as a domain controller, the installer decides whether to add the machine to an existing domain to create a new one.

The ArtiSan server has a role known as a member server. The distinction between a member server and a member workstation is simply determined by whether the OS on the computer is the server or workstation version of NT. There are no material differences in security between the two.

Domain Controllers

In the days of NT, domain controllers were organised in a hierarchy, the master being known as the PDC (primary domain controller) and the slaves as BDCs (backup domain controllers). The domain was the top level of the hierarchy. Companies with multiple domains used trust relationships to allow interoperation between the domains. Trust relationships are difficult to implement reliably and so management at an Enterprise level was quite hard to achieve. Windows 2000 introduced the concept of Active Directory Servers to replace the old domain system.

Active Directory provides a secure, structured, hierarchical storage of information about the interesting objects in an enterprise network; such as users, computers, services, and so on. The directory provides support for locating and working with these objects.

The directory system allows you to organise a directory of principals and other information within the domain and to make it accessible to other members of the domain. Domains are typically organised along the lines of separate Organisational Units (OUs). Each OU represents some subpart of the domain that has a common set of policies in force. In addition, the directory allows the administrator to express relationships between those principals such as the membership of groups (roles). In Active Directory, a group is simply a role that a user has within the security system. Users can belong to as many groups as makes sense for the organisation to function. The directory also allows the administrator or other responsible person to associate information with the principals such as mail addresses, postal addresses and so on.

Active Directory also provides a directory access service (using standard LDAP) to allow clients access to the directory information such as looking up the email address of another user in the domain. Like all NT resources the directory is also a resource with permissions associated with it. For instance, while it might be reasonable to allow one user of a domain to look up another's email address, it would be entirely unacceptable for that user to be able to look up their home address. Again note that a principal might be a computer. For instance, not only is ADS used to hold all the directory information used by Exchange, it also includes information associated with the computers that the Exchange Server software runs on. In fact, the Active Directory is the key resource for domain wide information. Active Directory also holds and distributes the policies that apply to a domain as a whole.

Active Directory is also a security database. When you convert an NT server to be an Active Directory domain controller, the local security database is disabled on the computer and replaced with a domain security database commonly known as Active Directory. This is why you cannot edit the local security database of the Active Directory Server: the local security database is disabled. It is also the reason that some built-in accounts, such as the Local System account, are no longer visible on an Active Directory controller.

Depending on the size of the organisation, you might have one or more Active Directory servers replicating information for redundancy. Active Directory largely discards the old master/slave relationship of NT domain controllers and replaces it with a peer-to-peer relationship. This provides greater reliability in the system. If a single server fails, another has a copy of the directory and takes over automatically. Information is replicated between Active Directory servers using RPC (or via SMTP if the routing architecture of the organisation does not allow direct connection using RPC).

Even larger organisations may be maintaining many such domains organised as a tree of domains or a forest of domains that have trust relationships between them. The distinction between a tree and a forest is to do with contiguity of DNS space. Both trees and forests of domains share a common schema and global catalog. One purpose of having this scalable structure is to allow administration to be delegated throughout the organisation.

Because the Active Directory is accessed through the network, it is important that the directory is accessible to the member computers. To achieve this, an organisation typically runs a DNS server integrated to Active Directory. This is a special DNS server that knows that when it cannot resolve a request, it should forward it to another domain server. It also supports DDNS (Dynamic DNS) that allows member computers to register their DNS name with the ADS on connection to the domain. All member computers are configured to use the ADS-integrated DNS and on boot up, register their DNS names with the Active Directory. This is handy because it means that the DNS addresses for member computers are visible to others through DNS. If the member server uses another DNS provider, all sorts of basic operations fail because the member computer cannot access the domain controller.

Active Directory is supplied with standard implementations of other services such as DHCP, WINS and so on. Smaller organisations co-locate all these services on a single box for simplicity. Active Directory can be used as a repository for domain-wide application services as well. For instance, there is an option in SQL Server to register itself with the Active Directory and can provide services via that directory. In the long term, ADS will also function as “naming service” on the network allowing domain members transparent access to named business functions.

ArtiSan cannot be installed on an Active Directory domain controller.

Domain Members

All computers that are not Active Directory controllers are referred to as domain members. A domain member might be a user workstation or a server providing specific application or network resources. For instance, Exchange is, typically, installed on a domain member server. NOTE: it is also possible to install Exchange on the domain controller allowing Microsoft to address the needs of the small business who may only have a single network server that needs to operate in a multiplicity of roles. ArtiSan must be installed on a domain member server, but must NEVER be installed on a machine running Exchange.

All domain members actually have two separately accessible security databases available at all times, the database on the domain controller and the local security database. This is an important distinction between a member server and a domain controller. Adding a computer to a domain does not disable the local security database, but rather adds an additional security database to the system and makes it the default. Adding a member to a domain requires some adjustments to services such as DNS so that the domain member can reliably communicate with the domain controller and with other domain members.

For our examples, we will assume that we have a domain called FIRM and a member computer called ArtiSanSERVER. In NT, the fully qualified name of a principal is in the form:

SECURITY_DB\PRINCIPAL_NAME

A user might be specified as “ArtiSanSERVER\IWAM_ArtiSanSERVER”. This means that the user is authenticated using the local security database on the ArtiSanSERVER host and the name of the principal is IWAM_ArtiSanSERVER. Equally, another principal might be the group “FIRM\Account Operators” which means the group “Account Operators” that exists in the domain FIRM. As stated above, when authenticating you can sometimes get away with not supplying the security database and NT will use the default. However, you should NEVER rely on this when configuring software because it does not always work.

If you view the permissions for a file and click “Add” you are presented with a dialog that allows you to add users either as defined in the local security database (ArtiSanSERVER\User) or from the domain (FIRM\User). It is important to understand that accounts with the same name in these two databases are entirely different accounts. For instance, the ArtiSanSERVER\Administrator is a different account from FIRM\Administrator. We normally refer to the former as the local administrator and the latter as the domain administrator.

Another important issue is the naming of domains. This has caused a good deal of confusion because some software allows you to be relatively imprecise about naming a domain or user and other software does not. The consequence is that sometimes the wrong name is used and this causes a lot of confusion.

For instance, in Windows 2000, you can often get away with using a domain DNS name (e.g. “firm.com”) instead of a domain name. For instance, the user “fbloggs” might be specified as “firm.com/fbloggs”. GINA, which displays the standard NT logon prompt, will accept this sort of thing. In other cases, other standard services will resolve an SMTP email address to a given user, e.g. “fbloggs@firm.com”. Exchange does this a lot. However, other software such as the DotNet framework does not recognise either of these combinations. This is generally because there is a new API that allows this kind of specification, but not all software is using it yet. Some developers use new APIs the moment they are available and others wait a good few years before risking it! In general, you should avoid using these types of specification (at least until the usage is consistent) and rely on the actual domain name (sometimes called the NetBIOS name) in the form “FIRM/fbloggs”.

Rights and Privileges in Operation

Having digressed into the world of domain controllers and members, we can return to the view of security as seen by processes running on a domain member.

When a process is run, it is run under an account. The rights of the account are applied to the action of a process restricting what it can do according to the ACEs associated with the resources it interacts with. Equally, the OS enforces the privileges that are associated with that account. In a typical system, there are many processes operating under different accounts at once. In general, system services operate at a higher level of privilege than user processes. A key principal of security is that all processes should operate with the bare minimum of privilege to function and no more.

Impersonation

Not only are there many processes operating under different accounts running in a system, individual processes can change the account they run under on the fly. Sometimes they do this simply to temporarily acquire additional privileges enforcing the notion of “least privilege” stated above. At other times they do this because they are performing the operations on behalf of another user and are trying to do so with the privileges and rights of that user. This is a process known as **impersonation**. For instance, when IIS runs, it runs under an account called IWAM_COMPUTERNAME. This is an account with lots of

privileges and lots of rights. When a user browses to the Web Server in the IIS, the IIS temporarily impersonates the client's account losing all the privileges that it had, performs the requested action at the lower security level and then reverts back to the IWAM_ account. This is a very common process in NT.

Let's suppose that there is a web site under the IIS that is marked as allowing anonymous access. When a user connects, the IIS notices the anonymous access and immediately impersonates the IIS Guest Account called IUSR_COMPUTERNAME. This account has very few privileges and rights and so restricts the actions the user can take. Alternatively, if the web site is marked for Integrated Windows Authentication, the IIS immediately challenges the Web Browser. When it issues the challenge it announces a list of the authentication protocols that may be used. The browser chooses an appropriate protocol and provides credentials. Depending on browser configuration, it might simply send the current NT login credentials or it might prompt the user for those credentials (pops up a dialog). The web server validates the credentials with the appropriate SSPI and either issues another challenge or impersonates that user.

When a process impersonates a user, it can do so with various levels of impersonation. These are known as Anonymous, Identify, Impersonate and Delegate. Anonymous means do nothing. Identify means identify the client but do no more. Impersonate the client means pretend to be the client but only on the local box only and delegate means take on all aspects of the rights of the client even in interactions with other hosts in the network.

The level chosen depends on the process, the configurations of the system and the facilities offered by the SSPI. The NTLM security provider can do all the levels except Delegate. Kerberos can do all of them. The reasons for this are too complex to go into here, but are associated with the fact that the credentials supplied to the server in NTLM are incomplete. This is important because if the SSPI does not provide delegate rights, then the server cannot access any remote resources on behalf of the client. This is called the "Delegation issue". It is an issue because in NTLM a process on a server such as IIS impersonating a client cannot, in turn, access other computers in the network as that client. In Kerberos, they can, but only if the Administrator explicitly allows it. You see this as settings in ADS such as "Trust this computer for delegation".

It is important to understand that it is not just IIS that does impersonation. For instance, when you map a network drive, this is achieved using impersonation. Impersonation is done at all sorts of levels in the system. ArtiSan uses impersonation to achieve some of its functionality.

It is also important to realise that the various functions of ArtiSan are not simply executing in a single account, but may be executing in multiple accounts at the same time. The corollary of this is that the system is designed to operate assuming that the various rights and privileges are present to allow it to operate across multiple accounts.

User Tokens

As stated above, all operations are executed in the context of a user account. When a program logs in to a user account, the system presents it with an object called the **User Token**. The token is actually a handle to a structure held in the kernel that caches things like the group membership of the user, the set of privileges that the user has been granted, the kind of token (anonymous, identity, impersonate, delegate) and other information. This allows the kernel to perform checks on rights and privileges very quickly.

Once a process has acquired a token, the token can then be associated with the currently executing thread (assuming the process has sufficient privilege). Once this is done, all subsequent operations are performed using that token to evaluate the permissions. For instance, imagine a web server is set up to use Basic Authentication. This means that when a client attempts to browse to the server, the server will challenge the user for credentials. The browser will prompt the user for a SAM account name and password. Because we are using Basic authentication, the credentials are passed back to the server in clear text. The web server then executes a login as that user and, if it is successful, is given back a token for the user account. Now, it associates the token with the client connection. Whenever, it executes code on behalf of the client, it sets the token in the current thread.

In other words, it impersonates the user. When the operation is complete, it reverts to its own identity ready for the next request. Note, that a token is used on a per-thread basis, so it is possible for a process to be impersonating many different clients at once.

In addition to threads having a token associated with it, each process in NT also has a token. You cannot change the process token, it is assigned when the process is created. The only purpose of the process token is to initialise the thread token when a new thread is created. For instance, when NT executes a service, the Service Control Manager (SCM) interrogates the registry settings for the service to determine the account that should be used for the service. It then uses the credentials it has to Login as that user, receiving a token in return. It then impersonates the user (by setting its thread token) and then creates the process. The token is copied from the thread in the SCM to the new process being created. The SCM then reverts to its default identity (actually the LocalSystem account). Once the new process is created, the kernel constructs a default thread for that process and copies the token from the process to the thread and then executes the thread. The consequence of all this is that the service process executes as the user configured in the SCM. NOTE: this is a simplification but is essentially correct. This is how ArtiSan expects the Scheduler and the Requestor to be run.

ArtiSan takes this process further. When ArtiSan needs to access the Exchange, it creates a RequestProxy object. However, this can happen when running as an arbitrary user being impersonated in the back end of the web server. However, ArtiSan needs to be sure that the RequestProxy is running as the ArtiSan user in order to get access rights to the Exchange. So, the ArtiSan code asks the Requestor service to create a RequestProxy on its behalf. The Requestor service runs as the ArtiSan user and, because it is responsible for spawning the RequestProxy, so does the latter.

How are Privileges and Rights Checked?

NT does its various security checks at the point when an operation is requested. In NT, there are two basic types of operation, those that require specific privileges and those that do not. For instance, the LogonUser operation requires a specific privilege to be granted (actually there are two separate privileges required in W2K and three in W2K3). Alternatively, the CreateFile operation requires no specific privileges, but does require certain access rights to be completed successfully. Typically, the API documentation describes with each function whether there are special privileges required.

Suppose that the software wished to reboot the operating system. The API function to achieve this is ExitWindowsEx(). A quick glance at the documentation for the API states that in order to shut down or restart the system, the calling process must use the **AdjustTokenPrivileges** function to enable the SE_SHUTDOWN_NAME privilege. To paraphrase, the calling thread must be running in an account that has the privilege to shutdown the system and that privilege must be enabled. The important point here is that there is a contract in force. The application developer is required to write code that attempts to acquire the privilege to do the shutdown, but it is up to the system configuration to determine whether that privilege is granted or not.

The process of checking user access rights is also performed at the point of operation. For instance, when you open a file with write access, NT:

- a) Retrieves the current User Token from the executing thread.
- b) Retrieves the current user SID and the SIDs for the groups that a user is a member of.
- c) Retrieves the ACL for the resource. At this point it checks inheritance and, if set, aggregates the ACEs for the parent into the ACL.
- d) Walks the ACL looking for Deny ACEs that specify the user (either directly or through group membership). If the ACE is present and the rights required are present then the operation is denied (because the ACE is a Deny ACE).

- e) Rewalks the ACL looking for Allow ACEs that specify the user (again either directly or through group membership). If the ACE is present and the rights required are present, the operation is allowed.
- f) If no matching ACE is present or there are none that matches the required rights the operation is denied.

In reality, much of the information is cached, so the operation is very quick.

Sometimes, the actions are not as they might appear to be, often because the effect of system privileges overrides the effect of access rights. Imagine you have a directory that allows "List Children" only to Domain Admins and it has a child directory that allows access to the ArtiSan user. A question might be "how can the ArtiSan user access the child directory, if the parent does not allow it to list children?" The simple answer to this is a privilege called "Bypass Traverse Checking". This privilege is, typically, enabled to everyone because checking the permissions on all directories is expensive. The consequence is that the software does not check the permissions of the parent resource. In very secure environments, it can be disabled and result is that the ArtiSan user in our example would not be aware of the child directory let alone read its contents.

ArtiSan Rights and Privileges

In order to understand in greater detail what is happening in an ArtiSan system, a detailed understanding is required of the various accounts that the ArtiSan code executes in. For the most part, ArtiSan processes operate using the ArtiSan user account that is created in the initial steps of the install. Many of the install steps are geared to acquiring the appropriate rights and privileges for that account.

However, parts of the ArtiSan system run in the impersonated account of the end-user accessing the system. Consequently, some of the install steps are designed to give more general access to specific resources. For instance, when a user clicks on a stub to access a mail from the archive, the code in the ArtiSan server is impersonating that client. The access rights required to view the item are evaluated in terms of the user attempting to access the resource. If a user makes a request to access someone else's mail, they will be denied access to it. This is proper behaviour when accessing machine resources.

The above has additional knock-on effects. Suppose, the code needs to retrieve a configuration from the registry in order to access the users email (it always does), then it must have appropriate rights to read the registry value. If the ArtiSan code is running impersonating the client, then the registry access rights must be set up to allow that user to access the configurations. Consequently, there is a step in the install that requires that an additional access right to read be granted to Authenticated Users for certain registry keys. Equally, the same code may want to place log output in the ArtiSan.log file. Again, rights need to be granted to Authenticated Users to write to the log file or some other files in the same directory. Again, there is an install step to allow this to happen.

The notion of code executing in multiple accounts is key to understanding what the ArtiSan install is attempting to achieve.

In addition, ArtiSan management UI is a DotNet application. DotNet applications, by default, execute using the ASPNET account (or in W2K3, the NETWORK SERVICE account). In either case, the principle is the same. When you connect to the ArtiSan management UI, the code immediately does two things. Firstly, it authenticates the client to ensure that the client is a member of the ArtiSan Managers group in the domain. Only members of the ArtiSan Managers group can access the UI. Secondly, it impersonates the ArtiSan user. It does this to get sufficient rights to access various network and local resources. Since the ASPNET account needs to impersonate the ArtiSan account, it needs sufficient privileges to logon and impersonate that user. Some of the steps in the install process are geared to getting those rights for the ASPNET account.

The ArtiSan Install Process

The following section provides some explanatory notes concerning the purpose of the operations undertaken in the installation of the software. Note that I have missed any instructions that are not explicitly to do with the security system. I am using the 1.1.0.42 install guide as a basis for this because it has the most stringent requirements for security.

Operations on the Domain Controller

Create a User account and an ArtiSan Managers group in Active Directory

The purpose of this section is to create the domain account for the ArtiSan user and to set up a pair of groups. The ArtiSan user has to be a domain account because it needs to access domain resources such as mailboxes in the Exchange system. We specify having an account with a password that will not expire or require change because this would involve reconfiguring the software each time it changes.

This section also has the creation of the ArtiSan Managers group, which is used to control access to the ArtiSan Management UI. We place the ArtiSan account in that group so that the UI will grant access during the install process (which is carried out while logged in using the ArtiSan account). If you do not set this up, you will be denied access to the ArtiSan management UI.

The ArtiSan Reviewers group is a group of people who have access to all items in the ArtiSan repository for business purposes. If you do not set up this account, you will receive warnings in the ArtiSan log about an inability to resolve an account name. The name of this group is a configuration in the ArtiSan system.

Operations on the Exchange Server(s)

Enable Access for ArtiSan to the Exchange Server

The purpose of this section is to give the ArtiSan user access to everyone's mailboxes. We add the account with full control to the server and inheritance should make it propagate down to the various stores and mailboxes. If the user has disabled propagation, then the account needs to be added at each individual store level. If you do not do this, ArtiSan will report all sorts of permissions errors when it tries to access any account other than its own.

Create an Organizational Forms Library

From a security perspective, the purpose of this operation is to allow the ArtiSan user account to add forms to the Exchange Organisational Forms Library. If you do not perform this operation, you will be denied access to the library later in the install when you try to add the form from Outlook.

Exchange Server Throttling

The purpose of this section is to allow the ArtiSan user access to performance information in the Exchange server. ArtiSan uses this information to back off from the Exchange when it is busy. Failing to do this will result in possible effects on other users in the system such as timeouts being reported when end-users try to login to the Exchange.

In fact, this section has a number of security related operations. To access the performance data, you have to be able to:

- a) Access the file that holds the performance information (PERFCXXX.DAT)
- b) Access the file that contains the names of the performance counters (PERFHXXX.DAT) as this tell the software how to index the file in a)
- c) Access the registry key that maps the files to a registry hive on the Exchange Server
- d) Access the service that publishes the performance information across the network,
- e) Access the RPC system that authenticates access to the service in d).

As we said above, there are many levels of access required to successfully access resources across the network.

The next few sections of the manual deal with installing and configuring dependent software.

Operations on the ArtiSan Server (as the Administrator)

Place ArtiSan User Account in Administrators group

This is used as a short cut. Many of the permissions required by the ArtiSan user account are granted by default to the local administrators. But, note that this must be the local administrators group not the Domain Admins group. This is because Domain Admins are explicitly denied access to some portions of Exchange and, as we stated above, Deny ACEs take precedence over Allow ACEs.

One side effect of this operation is that the ArtiSan user is given some specialised privileges on the local box such as:

- Act as part of the operating system
- Impersonate a client after authentication
- Log on as a service

- Log on as a batch job

However, it is possible that a domain administrator has restricted these rights at a domain level using GPOs. If so, you will need to ask the administrator to relax those restrictions for the ArtiSan server or the ArtiSan account. There are number of ways to achieve this depending on the knowledge/skill/sophistication of the administrator and their relative degree of paranoia. One approach is to add additional policy settings at the domain level. Another is to move the ArtiSan server into a separate organisational unit and to block policy inheritance to that organisational unit.

The rest of the install is carried out whilst logged into the ArtiSan user account. This can happen because we have added that account to the local administrators group.

Operations on the ArtiSan Server (as the ArtiSan user)

Enable Access to the Archive Store

As noted above, if the archive store is remote from the ArtiSan server, then we cannot access it using ordinary account credentials because we cannot guarantee “delegate” rights whilst we are impersonating the end-user. In pre-3.0 versions, if the store is remote, we place the ArtiSan system in “EnableDelegation” mode (the second part of this instruction). In later versions Delegation Mode is always turned on as part of Advanced Permissions. In this mode, all archive access occurs via the RequestProxy that operates in the ArtiSan account. However, we still need to be sure that the ArtiSan account has access to the Remote Store. At this stage we should check both the file share permissions and the NTFS permissions on the store. In general, the store permissions should be set up to allow access to the ArtiSan account.

Creating & Enabling the ArtiSan Web Service

The ArtiSanWeb site is intended to impersonate end-users who attempt to access it to view or restore mail. Consequently, ArtiSanWeb must disable anonymous access because we need the system to have the ability to impersonate the user. Also, we do not allow basic authentication simply because it is wise to disallow clear text passwords being transmitted across the network. If you do not do this, the web server will operate in the very restrictive IUSR_COMPUTER IIS guest account and will probably not get any access to the archives. The end effect is that when the user attempts to view a mail, they will be told that access is denied.

Installing the ArtiSan Database

ArtiSan maintains the login to the ArtiSan database using SQL server login, not Windows Integrated login. The reason for this is to allow end users to maintain the SQL server separate from the domain. If this step is not performed, you will get lots of log warning about not being able to access the database.

Create a System DSN to ArtiSan Database

This step has no effect on the configuration of the system. The only purpose if using credentials in this step is to allow the ODBC wizard to access the database server to fill out the various dialogs such as the list of databases on the server. The credentials are NOT stored anywhere as a result of this operation.

Modify the Web.Config file

The ArtiSan Manager software is an ASP.NET application. There are two important security aspects to this, authentication and impersonation. The first modification to this file is required to restrict access to only members of the “ArtiSan Managers” role on the domain. The requirement is for the installer to specify the domain name in question. ASP.NET does not assume the default domain when testing the credentials of the user. If you do not do this you will get an error message referencing the line of the Web.Config file indicating that the role is not known.

The second step is to configure the impersonation of the application. The ArtiSan manager web site always operates while impersonating the ArtiSan user. In order to do so, it needs

the credentials of that user account so that it can successfully login. If you do not do this, there will be an error indicating that the login as the ArtiSan user has failed.

Configure Identity Impersonation (optional)

The problem with the above approach is that the Web.Config stores a plain text password. This may be unacceptable to the customer. The next section illustrates how to set up the system to encrypt the password in the registry. This is the only change that is made by the “Configure Identity Impersonation” section.

The steps in this procedure use a standard Microsoft utility called aspnet_setreg to encrypt and store a set of user credentials in the registry. You then modify the Web.Config file to tell ASP.NET to read and decrypt the credentials from the registry when the application is invoked. The problem with this procedure is that you then need to grant the ASPNET account access rights so that it can actually read the encrypted information from the registry. If you do not do this you will get an error claiming that the registry cannot be read.

Set User Permissions

As stated above, ArtiSan software is often run while impersonating the account of the end-user. The consequence of this is that the software needs to be able to read registry configurations and to be able to write to the ArtiSan log file. If you do not do this you will get a lot of errors because the system cannot read configurations such as the name of the domain server. Worse, the system may not have access to the data directory to report the errors in the log file, so you may not actually see these errors.

Enable Remote Store Permissioning

This section is intended to deal with an indexing issue when the store is held in a remote location. Microsoft Indexing Service automatically filters documents on the basis of the access rights associated with the underlying files. The problem is that it only does this filtering if the store is local. If it is remote, it does not perform filtering. The result is that when end-users perform searches they see all the documents in the store including other users email. To get round this problem, we have supplied the Permissioning EML software. This software adds an additional property to the content index that contains a list of the principals who can access the item. The ArtiSan search service detects that there is a remote store (using the EnableDelegation key described above) to amend users queries to take into account this additional property causing the filtering to occur correctly. If you do not do this, then all users in the system will be able to see other users email.

Registering the ArtiSan Services

The only security implication of this step is that the ArtiSan services (Scheduler and Requestor) retrieve the ArtiSan user credentials from the registry. Consequently, you must successfully configure the system prior to performing this step. If you change the credentials for the ArtiSan user (such as the password) then you must redo this step. If you do not, then the service control manager will report an error when you try to start the services.

Set the Local Security Policy

The purpose of this section is to get sufficient privileges to allow the ArtiSan management UI to be able to impersonate the ArtiSan user account and to allow the ASPNET backend process to run as a service. If you do not do this, you will get an permission denied error when trying to run the ArtiSan management UI.

Internet Explorer Permissions

The ArtiSan system makes considerable use of Web Protocols to access archive data. Generally, users employ Internet Explorer to access data. When users access the ArtiSan system they must authenticate to allow the ArtiSan server to validate there access rights to the store. The way this authentication works depends on a set of Internet Explorer configurations.

In order to determine what MSIE will do, you need to identify the zone associated with the ArtiSan server. This can be viewed when you access the ArtiSan server from a client computer. The zone is indicated in the bottom right hand corner of the display. This should be set to "Local Intranet". If it is set to anything else, it indicates that the MSIE has not been correctly configured.

Sometimes MSIE is not very clever when trying to identify the zone. It achieves this by performing string matches on the name of the host specified in the URL. This can cause problems because there are a number of ways to refer to the ArtiSan server (e.g. localhost, short host name, fully qualified DNS name). In general, the best way to set up the ArtiSan system is to define an alias in DNS (as described in the install manual) and to configure the ArtiSan system to use that fully qualified name.

The zone determines the behaviour of the MSIE according to the settings associated with that zone. MSIE can be configured in the zone settings to automatically login to a web site. One of the easiest ways to configure MSIE is to use the default domain GPO. To do this, you get the administrator or the system to open the Group Policy Editor for the domain, you navigate to User Settings and you will find that you can specify both:

- a) The list of machines that are members of the Local Intranet zone. You should add the ArtiSanSERVER (using the name configured in the ArtiSan management UI). It is also common to configure the Local Intranet zone at this level using a wildcard (*) along with the local DNS name of the domain such as *.firm.com. The result of this is that all machines in the domain are automatically recognised and in the Local Intranet.
- b) The settings associated with the local intranet zone. You should set this to login automatically using the current users credentials.

The policy will cause these settings to be propagated to all domain computers. As a result, end-users will not need to authenticate themselves when accessing the ArtiSan web site.