Access mask constants

TAccessItem

The following constants should be used to set value of property TAccessItem. Mask.

These constants apply to TNTShare and TNTFileSecurity instances.

famRead	read access on the object is granted, denied or watched over
famWrite	write access on the object is granted, denied or watched over
famExecute	read right on the object is granted, denied or watched over
famDelete	delete right on the object is granted, denied or watched over
famPermissions	right to change permissions on the object is granted, denied or watched over
famOwnership	right to take ownership of files is granted, denied or watched over
famFullControl	all the listed rights are granted, denied or watched over

The following constants are responsible for inheritance behaviour of folders

genericRead	files created in the directory will have read access
genericWrite	files created in the directory will have write access
genericExecute	files created in the directory will have execute access
genericAll	files created in the directory will have full access

These constants apply to TNTRegSecurity instances.

kamQueryValue	access to read key value is granted, denied or watched over
kamSetValue	access to write key value is granted, denied or watched over
kamCreateSubKey	access to create subkey is granted, denied or watched over
kamEnumSubKey	access to enumerate subkeys is granted, denied or watched over
kamNotify	access to notify when changes occurred is granted, denied or watched over
kamCreateLink	access to create links is granted, denied or watched over
kamDelete	access to delete key is granted, denied or watched over
kamWriteDAC	access to write security information is is granted, denied or watched over
kamWriteOwner	access to take ownership of key is granted, denied or watched over
kamReadControl	access to read security information is granted, denied or watched over
kamFullControl	full access to the key is granted, denied or watched over

TUserInfo.AccountExpires

property AccountExpires: TDateTime;

Description

Specifies the date when the account expires. A value of "-1" indicates that the account never expires.

TUserInfo.AccountExpires

property AccountExpires: TDateTime;

Description
Specifies when the account will expire. A value of "-1" indicates that the account never expires.

TNTService.ActiveManager

TNTService Example

property ActiveManager: boolean;

Use ActiveManager: property to establish a connection to the service control manager on the specified computer and to open the specified database.

TNTService.ActiveService

property ActiveService: boolean;

Use ActiveService property to open a handle to an existing service.

Note
At the moment of switching ActiveService to true the properties of TNTService component are automatically changed in accordance with the service control manager database. The changes made after ActiveService is turned to true affect the database.

Assigning security attributes to folders

TNTFileSecurity

The folder is an object that contains files. In other words it is container. It may have some attributes that should be inherited by files that are inside the folder. That's why the folder's access entries are often duplicated. The table below shows some of the standard combinations of "access control entries" that Windows NT explorer assigns and correspondent combination of .

Explorer option	Access Mask	Access Flag
List(RX)(Not specified)	famRead **or** famExecute;	[acfContainer]
Read(RX)(RX)	1. genericRead **or** genericExecute; 2. famRead **or** famExecute;	[acfObjInherit, acfInheritOnly] [acfContainer]
Add(WX)(Not specified)	famWrite **or** famExecute;	[acfContainer]
Add & Read (RWX)(RX)	1. genericRead **or** genericExecute; 2. famRead **or** famWrite **or** famExecute;	[acfObjInherit, acfInheritOnly] [acfContainer]
Change(RWXD)(RWXD)	1. famDelete **or** genericRead **or** genericWrite **or** genericExecute; 2. famRead **or** famWrite **or** famExecute **or** famDelete;	[acfObjInherit, acfInheritOnly] [acfContainer]
Full control (All)(All)	1. genericAll 2. famFullControl	[acfObjInherit, acfInheritOnly] [acfContainer]

The other combinations are possible but Explorer.exe will call them "special access" or may even refuse to edit access attributes at all. It does not mean that access control does not work. It works, but Microsoft Explorer.exe cannot recognize the combination.

TNTUserMan.BadPasswordCount

TNTUserMan TUserInfo

property BadPasswordCount: integer; read only

Description

Specifies the number of times the user tried to log on to the account using an incorrect password. A value of "-1" indicates that the value is unknown.

TNTService.BinaryPathName

property BinaryPathName: string;

Description

Contains the fully qualified path to the service binary file.

TUserInfo.CodePage

property CodePage: integer;

Description

Specifies the code page for the user's language of choice.

TUserInfo.Comment

property Comment: string

Description

The string contains the comment. Make sure having set valid UserName before using this property.

TNTShare.Connections

property Connections: TConnectionList;

Description

This property returns pointer to the list of network drivers of local computer. Remember that property MachineName does not affect the list of connections. It is always a list of network (redirected) drivers of local computer. Each time you use Connections property component rereads the list of connections. Therefore use it only to obtain pointer to the list or to refresh information. Use retrieved pointer for any other operations.

Contact information

TNTService, TNTEventLog, TNTUserMan, TNTShare, TNTFileSecurity, TNTRegSecurity, TNTPrivilege are the components that compound the powerful collection designed to use specific features of Windows NT™. This is the shareware version. Full version with source code and technical support is available for registered users.

E-mail	contact@risq.belcaf.minsk.by
Visit us at	http://www.belcaf.com
Register your version	Internet registration service

Contact information
Library hierarchy
Windows NT vs. Windows 95

 TNTService component

 TNTEventLog component

 TNTUserMan component

 TNTShare component

 TNTFileSecurity component

 TNTRegSecurity component

 TNTPrivilege component

TNTService.ControlService

TNTService Example

The ControlService function sends a control code to a Win32 service.

procedure ControlService(Code: TControlCode);

The function asks the service control manager to send the requested control code to the service. The service control manager sends the code if the service accepts the control and if the service is in a controllable state. All running services accept the SERVICE_CONTROL_INTERROGATE control code by default. Each service specifies the other control codes that it accepts.

TUserInfo.CountryCode

property CountryCode: integer;

Description

Specifies the country code for the user's language of choice.

TNTService.Createservice

TNTService Example

The Createservice procedure creates a service object and adds it to the specified service control manager database.

procedure Createservice;

Description

The Createservice procedure creates a service object and installs it in the service control manager database by creating a service name key in the registry with the following form:
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\ServiceName.
Information specified is saved as values under this key. Setup programs and the service itself can create any subkey under this service name key for any service specific information.

DBLockHandle property

TNTService

property DBLockHandle: SC_LOCK; **read only**

Property DBLockHandle contains lock to the specified service control manager database when DBLocked is true. There is no other need to use it directly. To unlock a service control manager database reset DBLockHandle to false.

TNTService.DBLocked

property DBLocked: boolean

Description

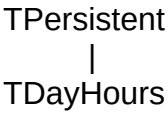
The DBLocked property allows you to acquire a lock on the specified database. Only one process at a time can have a lock on a database. A lock is a protocol used by setup and configuration programs and the service control manager to serialize access to the service tree in the registry. The only time the service control manager acquires a lock is when it is starting a service.

TNTService.DatabaseName

TNTService

property DatabaseName: string

Contains the string that names the service control manager database to open. This string should specify ServicesActive. If the string is empty, the ServicesActive database is opened by default.


TPersistent
TDayHours

TNTService.DeleteService

procedure DeleteService;

Description

The DeleteService procedure marks the specified service for deletion from the service control manager database. The database entry is not removed until all open handles to the service have been closed by calls to the CloseServiceHandle function, and the service is not running. A running service is stopped by a call to the ControlService function with the CONTROL_STOP control code. If the service cannot be stopped, the database entry is removed when the system is restarted. The service control manager deletes the service by deleting the service key and its subkeys from the registry.

Note

ActiveService property must be set TRUE before this function is called

TNTService.Dependencies

TNTService Example

property Dependencies: TStrings;

Description

Contains the names of services or load ordering groups that must start before this service. If the Dependencies is an empty list, the service has no dependencies. If a group name is specified, it must be prefixed by the SC_GROUP_IDENTIFIER (defined in the WINSVC.PAS file) character to differentiate it from a service name, because services and service groups share the same name space. Dependency on a service means that this service can only run if the service it depends on is running. Dependency on a group means that this service can run if at least one member of the group is running after an attempt to start all members of the group.

TNTService.DisplayName

property DisplayName: string;

Description

The string that is to be used by user interface programs to identify the service. This string has a maximum length of 256 characters. The name is case-preserved in the Service Control Manager. Display name comparisons are always case-insensitive.

UserInfo.Domain

property Domain: string **read only**

Description

Property contains the name of the the domain user belongs to.

TNTService.ErrorControl

TNTService

property ErrorControl: TErrorType;

Description

Specifies the severity of the error if this service fails to start during startup, and determines the action taken by the startup program if failure occurs.

Now event log has 3 sections: "Application", "Security", "System"

Event log sources

TNTEventLog

Event logging management information is stored in the **Services** key of the configuration database and can be modified by a system administrator.

The structure of the configuration information is as follows:

```
HKEY_LOCAL_MACHINE
  SYSTEM
    CurrentControlSet
      Services
        EventLog
          Application
          Security
          System
```

The EventLog key contains several subkeys, called *logfiles*. The default logfiles are **Application**, **Security**, and **System**. Each logfile subkey can contain subkeys, called *sources*. You cannot use a source name that has been used as a logfile name, and source names should not be hierarchical. (The backslash character [\] cannot be used in a registry key.) Each source entry contains information specific to the source of the event, as shown in the following table.

Value	Description
EventMessageFile	Specifies the path for the event identifier message file.
CategoryMessageFile	Specifies the path for the category message file. The event category and event identifier message strings can be in the same file.
ParameterMessageFile	Specifies the path for the event source's parameter message file. This file contains language-independent strings that are to be inserted into the event **description** strings. You can use the same message file for parameter, category, and event identifier message strings.
CategoryCount	Specifies the number of categories supported.
TypesSupported	Specifies a bitmask of supported types.

When TNTEventLog reads the description of events it searches for the specified source name in the registry. If the specified source name cannot be found in the registry, the **Application** logfile is used by default. However, because there is not a message or category string file, the event viewer will not be able to map the event identifier or category to a replacement string. For this reason, the recommended procedure is to add a unique source name for the application to the registry. This allows you to specify message files for the event identifier and category in your events. Applications and services should add their source names to the **Application** logfile. Device drivers should add their source name to the **System** logfile.

TNTEventLog component uses the **LoadLibrary** function to load the file indicated by the source's **EventMessageFile** registry value. The component then uses the **FormatMessage** function to retrieve the description string from the loaded module. To create message table in the DLL or EXE file you can use two tools. Microsoft C++ has a message compiler MC.EXE which allows to create binary resources. Those resource

can be added to the DLL or EXE later. Alternatively you can use MTEditor.exe which edits message tables resources inside EXE or DLLs. The source code of MTEditor.dpr is included in the full version of "Component Set for Windows NT".

Example

TUserInfo.FullName

property FullName: string;

Description

The property contains the full name of the user.

TNTService.GetDependentServiceList

TNTService Example

function GetDependentServiceList(AState: TServiceStates): TEnumList

Description
The GetDependentServiceList function returns the list of services that depend on the open service; that is, the specified service must be running before the enumerated services can run.

Note
ActiveManager property must be TRUE before this function called;
ActiveService property must be TRUE before this function called;
ServiceAccess property must include S_ENUMERATE_DEPENDENTS value;
function GetDependentServiceList returns an object, therefore you have to free it yourself;

TNTService.GetServiceDisplayName

TNTService See also Example

The GetServiceDisplayName function obtains the display name that is associated with a particular service name. The service name is the same as the service's registry key name.

function GetServiceDisplayName(AServiceName: string): string;

Note

ActiveManager property must be TRUE before this function called

TNTService.GetServiceKeyName

TNTService See also Example

function GetServiceKeyName(ADisplayName: string): string;

The function obtains the service name that is associated with a particular service's display name. The service name is the same as the service's registry key name.

Note
ActiveManager property must be TRUE before this function called

TNTService.GetServiceList

TNTService Example

function GetServiceList(AState: TServiceStates; AType: TEnumSevices): TEnumList;

Description

Function returns list of services that match the search condition
Parameters:
AState: define services in which state (running, stopped etc.) will be included in the result list
AType: type of services to be returned in the result list (drivers or/and processes)

Note

Returned value is an object and you have to free it when no more need.
ManagerAccess property must include M_ENUMERATE_SERVICE value before setting ActiveManage to true.

TUserInfo.HomeDir

property HomeDir: string;

Description

Property contains the path of the home directory for the user specified by UserName

TNTUserMan.LastLogOff

TNTUserMan TUserInfo

property LastLogOff: TDateTime; **read only**

Description

Specifies when the last logoff occurred. A "-1" means that the last logoff time is unknown.

TNTUserMan.LastLogon

property LastLogon: TDateTime **read only**

Description

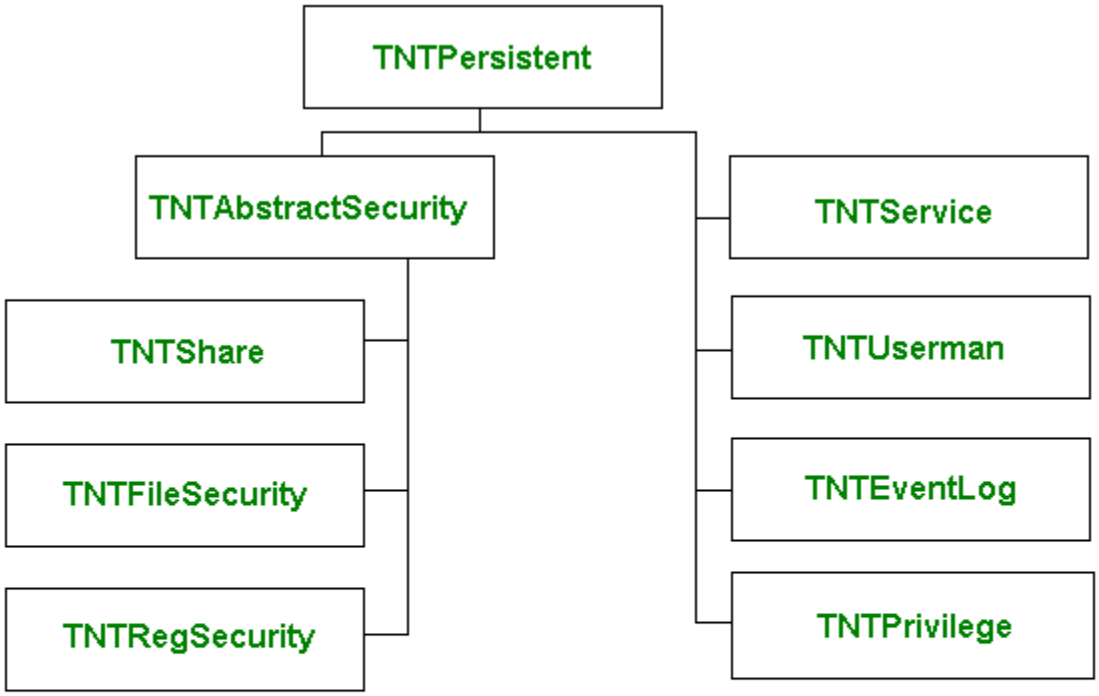
Property specifies the date and time when the last logon occurred. If the value is equal -1 it means that user never logged on.

Scope
Published

Accessibility
Read-only

Library hierarchy

NTSet's object hierarchy



TNTService.LoadOrder

property LoadOrder: string;

Description

The property names the load ordering group of which this service is a member. If the string is empty, the service does not belong to a group.
The registry has a list of load ordering groups located at HKEY_LOCAL_MACHINES\System\CurrentControlSet\Control\ServiceGroupOrder.
The startup program uses this list to load groups of services in a specified order with respect to the other groups in the list. You can place a service in a group so that another service can depend on the group.

The order in which a service starts is determined by the following criteria:

1. The order of groups in the registry's load-ordering group list. Services in groups in the load-ordering group list are started first, followed by services in groups not in the load-ordering group list, and then services that do not belong to a group.
2. The service's dependencies listed in the Dependencies property and the dependencies of other services dependent on the service.

TNTUserMan.LocalGroupComment

property LocalGroupComment: string;

Using

Windows NT only

The property will return empty string on Windows 95 machine

Description

This property is used to get and set description of local group either on local machine or network computer.

Remark

LocalGroupComment contains proper information only if LocalGroupName is valid group name.

TNTUserMan.LocalGroupMembers

TNTUserMan Example

property LocalGroupMembers: TStrings;

Using

Windows NT only

The property will return empty list on Windows 95 machine

Description

LocalGroupMembers property allows to retrieve and set list of members of particular local group on a local or remote computer. Use this property to replace the whole list of members as well as to add(remove) particular user into(from) a local group See also MemberOfLocal property. Each item of list is expected to be in the form of "DomainName\UserName".

Note

Before using this property make sure that LocalGroupName contains valid global group name.

TNTUserMan.LocalGroupName

property LocalGroupName: string

Using

Windows NT only

The attempt to set this property will not have any result on Windows 95 machine

Description

This property contains the name of the local group of users on the selected computer. Set this property before retrieving any other information about local group (see also LocalGroupComment, LocalGroupMembers). You may get the list of all local groups on the given computer using LocalGroups property.

TNTUserMan.LocalGroups

TNTUserMan Example

property LocalGroups: TStrings;

Windows NT only
The property will return empty list on Windows 95 machine

Description
TNTUserMan.Groups property contains list of local groups on a local or remote computer. Using this property you can retrieve list of local groups, add, delete group and replace the whole list of groups. Use LocalGroups' Add and Delete methods to add or remove global group. Before using this property make sure that MachineName contains valid computer name.

TUserInfo.LogonCount

property LogonCount: integer; **read only**

Description

Counts the number of successful times the user tried to log on to this account. A value of "-1" indicates that the value is unknown.

TUserInfo.LogonServer

property LogonServer: string; **read only**

Description

Property contains the name of the server to which logon requests are sent. Server names should be preceded by two backslashes (\\). When the server name is indicated by an asterisk (*), the logon request can be handled by any logon server. An empty string indicates that requests are sent to the domain controller.

TNTService.ManagerAccess

property ManagerAccess: TManagerAccess;

Description

Specifies the access to the service control manager. Before granting the requested access, the system checks the access token of the calling process against the discretionary access-control list of the security descriptor associated with the service control manager object. The M_CONNECT access type is implicitly specified by calling this function.

Note

ManagerAccess must be set before setting ActiveManager property to true.

TNTService.ManagerHandle

TNTService Example

ManagerHandle is a handle to the specified service control manager database

property ManagerHandle: SC_HANDLE; **read only**;

Description

Use ManagerHandle to call a Windows API function that requires the handle of a service control manager. Pass ManagerHandle as schSCManager parameter to these functions.

TUserInfo.MaxStorage

property MaxStorage: integer;

Description

Specifies the maximum amount of disk space the user can use. Use the value of "-1" to use all available disk space.

TNTUserMan.MemberOfGlob

property MemberOfGlob: TStrings;

Description

MemberOfGlob property contains list of global groups on the selected computer the user is member of. Use MemberOfGlob's Add and Delete
methods do change membership. Make sure that MachineName has a valid computer name and UserName contains valid user name before using this property.

TNTUserman.MemberOfLocal

property MemberOfLocal: TStrings;

Description

MemberOfLocal property contains list of local groups on the selected computer the user is member of. Use MemberOfLocal's Add and Delete
methods do change membership. Make sure that MachineName has a valid computer name and UserName contains valid user name before using this property.

TNTService.NotifyBootConfigStatus method

TNTService

The NotifyBootConfigStatus function notifies the service control manager as to the acceptability of the configuration that booted the system.

procedure NotifyBootConfigStatus(BootAcceptable: boolean);

BootAcceptable parameter

Specifies whether the configuration that booted the system is acceptable. If this parameter's value is TRUE, the service control manager saves the configuration that booted the system as the last-known good configuration. If the parameter's value is FALSE, the system immediately reboots, using the previously saved last-known good configuration.

TUserInfo.OperatorRights

property OperatorRights: TOperatorFlags;

Description

Property specifies the user's operator privileges.

TUserInfo.Options

property Options: TUserFlags;

Description
Contains values that determine several features of the account.

TUserInfo.Password

property Password: string;

Description

Property allows to set password for the user specified by UserName.
It returns set of asterisk when is read.

TNTService.Password

property Password: string;

The property contains the password to the account name specified by the ServiceStartName property, if the service type is WIN32_OWN_PROCESS or WIN32_SHARE_PROCESS. If the string is empty , the service has no password. If the service type is KERNEL_DRIVER or FILE_SYSTEM_DRIVER, this parameter is ignored.

Note
Make sure to change password before changing ServiceStartName.

TUserInfo.PasswordDate

property PasswordDate: TDateTime; **read only**

Description

The property represents the date when password was changed.

Priority of access control

All the components derived from TNTAbstractSecurity component have two lists which affect access rights on a given object (file, shared device, registry entry etc.). AccessAllowed grants access while AccessDenied revokes it. Generally, AccessDenied list has higher priority. This means that if you put an item into the AccessAllowed list which grants certain permissions to the user and put the same item into the AccessAllowed list, result will have prohibited permissions.

TUserInfo.Privilege

property Privilege: TUserPriv **read only**

Description

Use this property to determine level of permissions of user.

TNTService.QueryServiceLockStatus

TNTService Example

The QueryServiceLockStatus function retrieves the lock status of the open service control manager database.

function QueryServiceLockStatus: TQueryServiceLockStatus

Note

ManagerAccess must include M_QUERY_LOCK_STATUS value to call this function.
ActiveManager must be set TRUE.

TNTService.QueryServiceStatus

TNTService Example

The QueryServiceStatus function retrieves the current status of the open service.

function QueryServiceStatus: TServiceStatusClass;

Note

ActiveService property must be set before this unction is called;
AccessService property must have S_QUERY_STATUS access.
The return value is an object. Therefore you have to free it after getting information

TNTShare.Resources

property Resources: TResourceList;

This property returns pointer to the list of shared devices of computer specified by MachineName property. Each time you use Resources property component rereads the list of shared resources. Therefore use it only to obtain pointer to the list or to refresh information. Use retrieved pointer for any other operations.

TUserInfo.ScriptPath

property ScriptPath: string;

Description

It is the string specifying the path of the user's logon script, .CMD, .EXE, or .BAT file. The string can be empty.

Security object is a common name of all objects which can have security attributes. The examples of the security objects are: file, folder, shared device, registry key and so on. Each TNTAbstractSecurity descendant identifies an object in its own way:

TNTShare	by setting properties MachineName and ShareName;
TNTFileSecurity	by setting property FileName;
TNTRegSecurity	By setting properties MachineName, RootKey and CurrentPath;

AddEx

ControlService

GetServiceDisplayName

GetServiceKeyName

GetServiceList
GetDependentServiceList
TEnumList type

QueryServiceLockStatus

QueryServiceStatus

AccessAllowed
AccessDenied
SystemAudit

AddObject
IncludeSourceName
TEventType type

Service control manager database

TNTService

The services database includes information that determines whether each installed service is started on demand or is started automatically when the system starts. The database can also contain logon and security information for a service so that a service can run even though no user is logged on. It also enables system administrators to customize security requirements for each service and thereby control access to the service. No more than one instance of a service can be running at a time.

Services can be divided into these two groups: Win32 services that conform to the interface rules of the service control manager, and driver services that conform to the device driver protocols for Microsoft Windows NT™.

The service control manager maintains a ServicesActive database in the registry. This is the currently active database that was used to start the system. The following is the registry path to this database.

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services
In this directory, there is a ServiceName key for each installed Win32 service or driver service. The ServiceName key is the name of the service specified by the Createservice function when the service was installed.
The database includes the following information about each installed service:

The type of service. For Win32 services, the service type indicates whether the service executes in its own process or shares a process with other Win32 services. For driver services, it indicates whether the service is a kernel driver or a file system driver.

The start type of the service. The start type indicates whether the service is started automatically at system startup or whether the service control manager starts it when requested by a service control process. The start type can also indicate whether the service is disabled, in which case it cannot be started.

A fully qualified path of the service's executable file. The filename extension is .EXE for Win32 services and .SYS for driver services.

Optional information used by the service control manager to determine the proper order for starting services. This information can include a list of services that must be running before the service can start, the name of a load ordering group that the service is part of, and a tag identifier that indicates the start order of the service in its load ordering group.

For Win32 services, an optional logon account name in which the service process runs and an optional password for this account. If no account is specified, the service runs in the LocalSystem account.

For driver services, an optional driver object name (for example, \FileSystem\Rdr or \Driver\Xns), used by the I/O system to load the device driver. If no name is specified, the I/O system creates a default name based on the service name.

TNTService.ServiceAccess

TNTService Example

property ServiceAccess: TServiceAccess;

Description
Specifies the access to the service. Before granting the requested access, the system checks the access token of the calling process against the discretionary access-control list of the security descriptor associated with the service object.

Note
ServiceAccess must be set before setting ActiveService property to true.

TNTService.ServiceHandle

TNTService Example

ServiceHandle is a handle to the service.

property ServiceHandle: SC_HANDLE; **read only**;

Description

Use ServiceHandle to call a Windows API function that requires the handle of a service. Pass ServiceHandle as hService parameter to these functions.

TNTService.ServiceName

TNTService Example

Service name is associated with a particular service's display name. The service name is the same as the service's registry key name.

property ServiceName: string

Description

The maximum string length is 256 characters. The service control manager database preserves the case of the characters, but service name comparisons are always case insensitive. Forward-slash (/) and back-slash (\) are invalid service name characters.

TNTService.ServiceStartName

property ServiceStartName: string

Description

If the service type is SERVICE_WIN32_OWN_PROCESS or SERVICE_WIN32_SHARE_PROCESS, this name is the account name in the form of "DomainName\UserName", which the service process will be logged on as when it runs. If the account belongs to the built-in domain, ".\UserName" can be specified. If empty string is specified, the service will be logged on as the LocalSystem account.
If the service type is SERVICE_KERNEL_DRIVER or SERVICE_FILE_SYSTEM_DRIVER, this name is the Windows NT driver object name (that is, \FileSystem\Rdr or \Driver\Xns) which the input and output (I/O) system uses to load the device driver. If empty string is specified, the driver is run with a default object name created by the I/O system based on the service name.

TNTService.ServiceType

property ServiceType: TServiceTypes

Description

A set of bit flags that specify the type of service. You must specify one of the service type flags to indicate the service type. In addition, if you specify either of the SERVICE_WIN32 flags, you can also specify the SERVICE_INTERACTIVE_PROCESS flag to enable the service process to interact with the desktop.

TNTShare.Sessions

property Sessions: TSessionList

Description

This property returns pointer to the list of sessions established between a server specified by MachineName and workstations. Each time you use Sessions property component rereads the list of sessions. Therefore use it only to obtain pointer to the list or to refresh information. Use retrieved pointer for any other operations.

TNTService.Startservice

TNTService Example

The Startservice procedure starts the execution of a service.

procedure Startservice;

Description

When a driver service is started, the Startservice method does not return until the device driver has finished initializing. When a Win32 service is started, the service control manager spawns the service process, if necessary. If the specified service shares a process with other services, the required process may already exist. The Startservice method does not wait for the first status update from the new service (which may take a while). Instead, it returns when the service control manager receives notification from the service control dispatcher that the ServiceMain thread for this service was created successfully.

TNTService.StartType

property StartType: TStartType

Specifies when to start the service.

TAccessItem properties

TAccessItem TAccessList TNTFileSecurity

- UserName
- Mask
- Flags

TAccessItem type

TAccessItem represents an item in a TAccessList.

Description

A TAccessList holds a group of TAccessItem objects. TAccessItem may belong to the different TAccessList instances. When TAccessItem object belongs to the AccessAllowed list, it grants access to the users, specified by its UserName property on some system object (file, shared device, registry entry etc.). If the object is held by AccessDenied list it restricts access. Finally, items which belong to the SystemAudit list make the system to write to event log when a user performs the specified operation. TAccessItem objects are created and destroyed by TAccessList's Add, Delete and Clear methods. You will never need to create an instance of TAccessItem class explicitly.

Example

```
FileSecurity1.AccessList[0].Mask := famFullControl;
```

TAccessItem.Flags

TAccessItem TAccessList TNTFileSecurity

property Flags: TAceFlags;

Property indicates inheritance behavior of TAccessItem object. You need to use this property in two cases. If you are going to assign control access to the whole directory instead of single file - use acfContainer,acfInheritOnly, acfObjInherit, acfNoPropogate flags. If you want to watch over attempts to access file with SystemAudit property - use acfSuccAudit, acfFailAudit flags.

TAccessItem.Mask

TAccessItem TAccessList TNTFileSecurity

property Mask: DWORD;

Property specifies access which is granted, denied or watched over to the user(s) denoted by UserName property. The type of the action is implicitly defined by the owner of TAccessItem object. Access is granted when the owner is AccessAllowed list, denied when AccessDenied and watched over when SystemAudit. Use only predefined constants to be assigned to this property.

Example

FileSecurity.DeniedList[0].Mask := famRead **or** famExecute;

TAccessItem.UserName

property UserName: string;

Property contains the name of user or group of users that TAccessItem object affects. When writing this property you may specify string either in the form of "DomainName\UserName" or "UserName". The way in which TAccessItem object affects user(s) specified by UserName property depends on the list, the object belongs to. It may grant or restrict access as well as provoke system audit.

TAccessList methods

TAccessList TNTFileSecurity

- Add
- Delete
- Clear

TAccessList properties

TAccessList TNTFileSecurity

- Items
- Count

TAccessList type

Properties Methods TNTFileSecurity TNTShare

TAccessList is a container for TAccessItem objects. The rights granted or denied by each item are accumulated. The Count property contains the number of items in the list. Use the Add, Delete, Clear methods to add and delete access control entries. Usually you will not create instances of TAccessList class. Use AccessAllowed, AccessDenied, SystemAudit properties instead to obtain a pointer to the required list which is maintained by components derived from TNTAbstractSecurity component.

Example

The example applies to TNTFileSecurity component.

```
FileSecurity1.FileName := '\\moon\Admin';
if FileSecurity1.AccessAllowed.Count = 0 then ShowMessage('No users have access to \\moon\Admin');
```

TAccessList.Add

TAccessList TNTFileSecurity

Add adds new item to the end of the list

function Add(UserName: string; AMask:DWORD; AFlags: TAceFlags): TAccessItem;

Description

The result of adding the item depends on the list into which the item is inserted. Add new item to AccessAllowed list to grant an access on a file or directory to a user or a group. All the users which are not included in the AccessAllowed list will not have an access to the file. Add new item to AccessDenied list to revoke an access to the file from a user or a group. Add new item to SystemAudit list to make system maintain the log of access attempts. AMask defines actions which will be granted, denied or watched over. AFlags determines the inheritance behavior of the newly added item. To find out how contradictions between AccessAllowed and AccessDenied lists are resolved see priority of access control.

Example 1: Grant access on the file

```
FileSecurity1.FileName := 'c:\Program Files\Internet\List.htm';
FileSecurity1.AccessAllowed.Add('Everyone',famFullControl, [ ]);
```

Example 2: Revoke access from the folder

```
FileSecurity1.FileName := 'c:\Program Files\Internet';
FileSecurity1.AccessDenied.Add('Everyone',0, [acfObjInherit, acfInheritOnly ]);
FileSecurity1.AccessDenied.Add('Everyone',famFullControl, [acfContainer ]);
```

Example 3: Grant access on a registry key.

```
NTReg.Security1.RootKey := HKEY_LOCAL_MACHINE;
NTReg.Security1.CurrentPath := 'SOFTWARE\BelCAF\Wizard';
NTReg.Security1.AccessAllowed.Add('Everyone',0, [acfObjInherit, acfInheritOnly ]);
NTReg.Security1.AccessAllowed.Add('Everyone',kamFullControl, [acfContainer ]);
```

TAccessList.Clear

TAccessList TNTFileSecurity

procedure Clear;

Description

Clear deletes all the items from TAccessList list. The meaning of this action depends on the instance of TAccessList class. Deleting all the items from AccessAllowed list means that no one will have an access to the object specified by FileName. Deleting all the items from AccessDenied list erases list of users or groups with explicitly prohibited access to the object. Deleting all the items from SystemAudit list prevents system from logging access attempts.

TAccessList.Count

TAccessList TNTFileSecurity

Count is the number of TAccessItem objects in the list.

property Count: Integer; read only;

Description

Read Count to determine the number of TAccessItem objects in the Items array.

Example

```
FileSecurity1.FileName := 'd:\Private\Documents';
if FileSecurity1.SystemAudit.Count = 0 then
    ShowMessage('Directory  d:\Private\Documents has no audit control');
```

TAccessList.Delete

TAccessList TNTFileSecurity

Delete removes the item at the position given by the Index parameter.

procedure Delete(Index: integer);

Description

The actual meaning of deleting one item from the list depends on the instance of TAccessList class, deleted item belongs to. The deletion of an item from AccessDenied list removes explicit restriction for the user or group on using the file. Whether the user will have access to the file or not depends on the information from AccessAllowed list. The deletion of an item from AccessAllowed list revokes access to the given file from the user or group. Deletion of an item from SystemAudit list informs system that there is necessity to watch over certain actions any more.

TAccessList.Items

TAccessList TNTFileSecurity

Items is the array of object references.

property Items[Index: integer]: TAccessItem; **default**;

Description

Use Items to obtain a pointer to a specific TAccessItem object in the array. The Index parameter indicates the index of the object, where 0 is the index of the first object, 1 is the index of the second object, and so on. Use Items with the Count property to iterate through all of the objects in the list.

Example

```
var
  PList: TAccessList;
  i: integer;
begin
  FileSecurity1.FileName := 'c:\work\doc';
  PList := FileSecurity1.AccessDenied;
  for i := 0 to PList.Count - 1 do ListBox1.Items.Add(PList.Items[i].UserName);
end;
```

TAccessTypes type

TUsage TNTShare

Type
  TAccessType = (actRead, actWrite, actCreate, actExec, actDelete, ActAtrib, actPerm, actFindFirst, actGroup);
  TAccessTypes = set of TAccessType;

Value	Meaning
actRead	Permission to read data from a resource and, by default, to execute the resource.
actWrite	Permission to write data to the resource.
actCreate	Permission to create an instance of the resource (such as a file); data can be written to the resource as the resource is created.
actExec	Permission to execute the resource.
actDelete	Permission to delete the resource.
ActAtrib	Permission to modify the resource's attributes (such as the date and time when a file was last modified).
actPerm	Permission to modify the permissions (read, write, create, execute, and delete) assigned to a resource for a user or application.

TAceFlags type

TAccessItem TAccessList TNTFileSecurity

```
type
  TAceFlag     = (acfObjInherit, acfContainer, acfInheritOnly, acfNoPropogate,
acfSuccAudit, acfFailAudit);
  TAceFlags = set of TAceFlag;
```

Value	Meaning
acfContainer	The TAccessItem object is inherited by container objects, such as directories.
acfInheritOnly	The TAccessItem does not apply to the container object, but to objects contained by it.
acfObjInherit	The TAccessItem object is inherited by non container objects, such as files created within the container object to which the TAccessItem object is assigned.
acfNoPropogate	The acfObjInheri and acfContainer flags are not propagated to an inherited access control entries.
acfSuccAudit	Used with TAccessItem which belong to SystemAudit list to indicate a message is generated for failed access attempts.
acfFailAudit	Used with TAccessItem which belong to SystemAudit list to indicate a message is generated for successful access attempts.

TConnection properties

TConnection TConnectionList TNTShare

- LocalName
- RemoteName
- UserName

TConnection type

Properties TNTShare TConnectionList

TConnection represents an item in a TConnectionList.

Description

A TConnectionList holds a group of TConnection objects. TConnection objects are created and destroyed by TConnectionList's Add, AddEx and Delete methods. You will never need to create an instance of TConnection class explicitly. Use TConnectionList's Items property to get pointers to TConnection instances maintained by TNTShare component.

TConnection.LocalName

TConnection TNTShare

property LocalName: string; read only;

LocalName returns the name of a local device used for redirection, such as "F:" or "LPT1". LocalName is specified at the time of establishing connection by TConnectionList's methods Add and AddEx. This property is empty string if the connection does not use a device.

Example

```
function TForm1.CheckDrive(ADrive: string);
var
  i: integer;
  PConnections: TConnectionList;
begin
  PConnections := Share1.Connections;
  for i := 0 to PConnections.Count - 1 do
    if ADrive = PConnections[i].LocalName then
      begin
      ShowMessage('Drive '+ADrive+' is already in use');
      Break;
      end;
end;
```

TConnection.RemoteName

TConnection TNTShare

property RemoteName: string read only;

RemoteName specifies the network resource which is connected to. The string follows the network provider's naming conventions. LocalName is specified at the time of establishing connection by TConnectionList's methods Add and AddEx.

Example

```
var
  PConnections: TConnectionList;
begin
PConnections := Share1.PConnections;
if PConnections.Count > 0 then  Label1.Caption := PConnections[0].RemoteName;
end;
```

TConnection.UserName

TConnection TNTShare

Property UserName: string; read only

Property specifies a user name which was used to make the connection. The UserName is returned in the form of "Domain\User". Use TUserList's Add method to establish new network connection using default user name and password. Use TUserList's AddEx method to indicate user name different from default one.

Example

```
var
   PConnections: TConnectionList;
   i: integer;
begin
   PConnections = Share1.Connections;
   for i := 0 to PConnections.Count -1 do
        ListBox1.Items.Add('Connection to the resource '+PConnections[i].RemoteName+
     ' uses UserName '+PConnections[i].UserName);
end;
```

TConnectionList methods

TConnectionList TNTShare

- Add
- AddEx
- Clear
- Delete

TConnectionList properties

TConnectionList TNTShare

Items

Count

TConnectionList type

Properties Methods TNTShare Connections

TConnectionList is a container for TConnection objects. It holds the local computer's connection list. All currently active connections (not only those remembered in the registry) are in the list. The Count property contains the number of items in the list. Use the Add, AddEx, Delete, Clear methods to add and delete connections. Usually you will not create instances of TConnectionList class. Use TNTShare.Connections property instead to obtain a pointer to the list of connections maintained by TNTShare component.

Example

```
if Share1.Connections.Count = 0 then ShowMessage('Network drivers not found!')
```

TConnectionList.Add

TConnectionList TNTShare See also

Method adds a new device to the list of network drivers. In other words it creates a local driver redirected to network resource.

procedure Add(LocalName, RemoteName: string; AType: TShareType);

Adds *LocalName* device redirected to *RemoteName* network resource. You must explicitly specify the type of device. Method takes default user name and password to establish a connection. If procedure fails it raises an exception which explains an error (*LocalName* is already in use; *RemoteName* not found; and so on)

Example

```
Share1.Connections.Add('F:', '\\moon\office', stDisk);
Share1.Connections.Add('LPT2:', '\\PrintServer\HP5', stPrint);
```

TConnectionList.AddEx

TConnectionList TNTShare

Method adds a new device to the list of network drivers. In other words it creates a local driver redirected to network resource. Use this method to establish a connection to the network device using UserName and password different from default ones.

procedure AddEx(LocalName, RemoteName: string; AType: TShareType; UserName, Password: string);

Description

For the description of the first three parameters look at Add method's description.
UserName specifies the name of user to be used when establishing connection.
Password specifies a password for the given user name.

Example

```
Share1.Connections.AddEx('F:', '\\moon\office', stDisk, 'Domain guest', 'GuestN34');
```

TConnectionList.Clear

TConnectionList TNTShare

Clear breaks all the connections of local computer with network resources.

procedure Clear; override;

Description

Call Clear to remove all network drivers of local computer. Call Delete to remove particular one(s).

TConnectionList.Count

Count is the number of entries in the list.

property Count: Integer; read only;

Description

Read Count to determine the number of TConnection objects in the Items array.

Example

```
Label1.Caption := 'Network drivers: ' + Share1.Connections.Count;
```

TConnectionList.Delete

TConnectionList TNTShare

procedure Delete(AIndex: integer);

Use Delete method to break an existing network connection. Connection will be removed from the registry and will not be restored after system reboot.

Example

```
procedure Form1.btnClearClick(Sender: TObject)
var
  PConnections: TConnectionList;
begin
  PConnections := Share1.Connections;
  while PConnections.Count > 0 do PConnections.Delete(0);
end;
```

TConnectionList.Items

TConnectionList TNTShare

Items is the array of object references.

property Items[Index: Integer]: TConnection; **default;**

Description

Use Items to obtain a pointer to a specific TConnection object in the array. The Index parameter indicates the index of the object, where 0 is the index of the first object, 1 is the index of the second object, and so on. Use Items with the Count property to iterate through all of the objects in the list.

TControlAcceptedSet type

type

```
  TControlAccepted = (ACCEPT_STOP, ACCEPT_PAUSE_CONTINUE,
ACCEPT_SHUTDOWN);
  TControlAcceptedSet = set of TControlAccepted;
```

Description

ACCEPT_STOP
: The service can be stopped. This enables the CONTROL_STOP value.

ACCEPT_PAUSE_CONTINUE
: The service can be paused and continued. This enables the CONTROL_PAUSE and CONTROL_CONTINUE values.

ACCEPT_SHUTDOWN
: The service is notified when system shutdown occurs. This enables the system to send a CONTROL_SHUTDOWN value to the service. The ControlService function cannot send this control code.

TControlCode type

TNTService See also

type

TControlCode = (CONTROL_STOP, CONTROL_PAUSE, CONTROL_CONTINUE, CONTROL_INTERROGATE, CONTROL_SHUTDOWN);

CONTROL_STOP
: Requests the service to stop. The Service must be open with S_STOP access.

CONTROL_PAUSE
: Requests the service to pause. The Service must be open with S_PAUSE_CONTINUE access.

CONTROL_CONTINUE
: Requests the paused service to resume. The hService handle must be open with S_PAUSE_CONTINUE access.

CONTROL_INTERROGATE
: Requests the service to update immediately its current status information to the service control manager. The hService handle must be open with S_INTERROGATE access.

CONTROL_SHUTDOWN
: The ControlService function fails if this control code is specified.

TCurrentState type

TNTService

type

TCurrentState = (STOPPED, START_PENDING, STOP_PENDING, RUNNING, CONTINUE_PENDING, PAUSE_PENDING, PAUSED);

Description

STOPPED	The service is not running.
START_PENDING	The service is starting.
STOP_PENDING	The service is stopping.
RUNNING	The service is running.
CONTINUE_PENDING	The service continue is pending.
PAUSE_PENDING	The service pause is pending.
PAUSED	The service is paused.

TDayHours class

Hierarchy Properties Methods TLogonHours

The class represents one day. It defines the enabled/prohibited connection ability for each of 24 hours. TNTUserMan component creates seven objects of the class. You will never need to create instances of the class yourself. Note that the class deals with GMT time. If your local time is GMT+2 and you want to enable connection from 20 p.m. to 21 p.m. on Wednesday then apply Hours[18] property:

TDayHours methods

TDayHours

- Clear
- SetAll

TDayHours properties

- Hours

TDayHours.Clear

TDayHours

procedure Clear;

Description

The procedure prohibits the connection ability by setting false value for each of 24 hours of day. When user tries to connect to the computer during disabled time, they get message: "You account has restricted time for connection. Please try latter."

TDayHours.Hours

property Items[AIndex: integer]: boolean;

Description

The property allows to change the state of connection ability for each of 24 hours. Set true to enable the connection during specified hour and set false to prohibit it. The valid index values are from 0 to 23. Note that the property deals with GMT time. If your local time is GMT+2 and you want to enable connection from 20 p.m. to 21 p.m. on Wednesday then apply Hours[18] property:

TDayHours.SetAll

procedure SetAll;

Description

The procedure enables the connection ability by setting true value for each of 24 hours of the day.

TDriveType type

TNTFileSecurity

type

TDriveType = (dtUnknown, dtError, dtRemovable, dtFixed, dtRemote, dtCDROM, dtRamDisk);

Value	Meaning
dtUnknown	The drive type cannot be determined
dtError	The root directory does not exist.
dtRemovable	The disk can be removed from the drive.
dtFixed	The disk cannot be removed from the drive.
dtRemote	The drive is a remote (network) drive.
dtCDROM	The drive is a CD-ROM drive.
dtRamDisk	The drive is a RAM disk.

TEnumList type

TNTService

TEnumList is a class that represent list of TServiceStatusClass objects.
The functions GetServiceList and GetDependentServiceList use this class for returning information.

```
TEnumList = class
public
  constructor Create(AOwner: TObject);
  destructor  Destroy;
  procedure Delete(AIndex: integer);
  procedure Clear;
  property Items[AIndex: integer]: TServiceStatusClass
  function IndexOf(AServiceName: string): integer;
  property Count: integer;
  property Parent: TObject;
  function Add: TServiceStatusClass;
end;
```

TEnumSevices type

TNTService See also

type

```
TEnumSevice = (DRIVER, PROCESS);
TEnumSevices = set of TEnumSevice;
```

Description

PROCESS Enumerates services of type WIN32_OWN_PROCESS and WIN32_SHARE_PROCESS.

DRIVER Enumerates services of type KERNEL_DRIVER and FILE_SYSTEM_DRIVER.

TErrorType type

TNTService

TErrorType = (ERROR_IGNORE, ERROR_NORMAL, ERROR_SEVERE, ERROR_CRITICAL);

SERVICE_ERROR_IGNORE
: The startup (boot) program logs the error but continues the startup operation.

SERVICE_ERROR_NORMAL
: The startup program logs the error and puts up a message box pop-up but continues the startup operation.

SERVICE_ERROR_SEVERE
: The startup program logs the error. If the last-known-good configuration is being started, the startup operation continues. Otherwise, the system is restarted with the last-known-good configuration.

SERVICE_ERROR_CRITICAL
: The startup program logs the error, if possible. If the last-known-good configuration is being started, the startup operation fails. Otherwise, the system is restarted with the last-known good configuration.

SERVICE_NO_CHANGE
: The existing StartType value is not to be changed.

TEventItem type

The type represents an event item in the event log and returns detailed information about the event. The **Objects** property returns an information using this class. Most probably, you will never need to create instances of TEventItem yourself.

TEventItem.SetData

TEventItem

procedure SetData(AData: Pointer; DataSize: integer);

procedure SetData copies the variable referenced by AData pointer into inner data structure.
The length of data to be copied is specified by DataSize parameter.

TEventType

TNTEventLog

type

```
  TEventType = (EVT_SUCCESS, EVT_ERROR, EVT_WARNING,
EVT_INFORMATION,
                EVT_AUDIT_SUCCESS, EVT_AUDIT_FAILURE);
```

EVT_ERROR	Error event
EVT_WARNING	Warning event
EVT_INFORMATION	Information event
EVT_AUDIT_SUCCESS	Success Audit event
EVT_AUDIT_FAILURE	Failure Audit event

TFilterAccountSet type

TNTPersistent

type

TFilterAccountSet = **set of** TFilterAccount;

TFilterAccount is an enumeration type with following values

Value	Meaning
FLT_TEMP_DUPLICATE_ACCOUNT	Enumerates local user account data on a domain controller
FLT_NORMAL_ACCOUNT	Enumerates user account data on a server.
FLT_INTERDOMAIN_TRUST_ACCOUNT	Enumerates account data within a specified domain.
FLT_WORKSTATION_TRUST_ACCOUNT	Enumerates account data for specified workstations.
FLT_SERVER_TRUST_ACCOUNT	Enumerates data for specified server accounts.

TLogonAs class

TNTPersistent

TLogonAs class stores user name and password to use when connecting to the remote system. This is necessary when the user wants to use the account that differs from that one under which he/she is currently logged in.

```
type
  TLogonAs = class
  published
    property UserName: string;
    property Password: string;
  end;
```

TLogonHours class

Hierarchy Properties Methods

The class represents the list that refers to the days of the week. The list has seven objects of class TDayHours which hold information about allowed logon time for each hour of each day from Sunday to Saturday.

TLogonHours methods

TLogonHours

- Clear
- SetAll

TLogonHours properties

- Days

- Sun
- Mon
- Tues
- Wed
- Thur
- Fri
- Sat

TLogonHours.Clear

TLogonHours

procedure Clear;

Description

The procedure prohibits the connection ability by setting false value for each of 168 hours of week. When user tries to connect to the computer during disabled time, they get message: "You account has restricted time for connection. Please try latter."

TLogonHours.Days

property Days[AIndex: integer]: TDayHours;

Description

The property holds the references to the days of week and is useful for iteration through them. The valid values of *AIndex* parameter are from 0 to 6. Remember that Sunday is considered to be the first day of the week. Alternatively, you can use the properties from Sun to Sat to refer to separate days. The following two examples are equivalent. They prohibit user Guest to connect on the Sunday.

TLogonHours.SetAll

TLogonHours

procedure SetAll;

Description

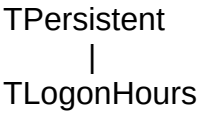
The procedure enables the connection ability by setting true value for each of 168 hours of week. The procedure calls SetAll method for each of seven TDayHours objects.

TLogonHours.Sun

property Sun: TDayHours;

Description

Each of properties from Sun to Sat represent one day of week from Sunday do Monday respectively. The properties refer to the objects of Days list. They define enabled/prohibited connection ability for each hour of day. Note that following examples are identical. They prohibit user Guest to connect on Sunday between 22 and 23 o'clock.


TPersistent
TLogonHours

TManagerAccess type

TNTService

type

TManagerAccess = set of TDesiredManagerAccess;
TDesiredManagerAccess = (M_CONNECT, M_CREATE_SERVICE, M_ENUMERATE_SERVICE, M_LOCK, M_QUERY_LOCK_STATUS, M_MODIFY_BOOT_CONFIG);

Description

M_CONNECT
: Enables connecting to the service control manager.

M_CREATE_SERVICE
: Enables calling of the Createservice function to create a service object and add it to the database.

M_ENUMERATE_SERVICE
: Enables calling of the EnumServicesStatus function to list the services that are in the database.

M_LOCK
: Enables calling of the LockServiceDatabase function to acquire a lock on the database.

M_QUERY_LOCK_STATUS
: Enables calling of the QueryServiceLockStatus function to retrieve the lock status information for the database.

TNTAbstractSecurity component

Hierarchy Properties Methods

The component is an abstract ancestor for other components that deal with Windows NT security. The component cannon be used without rewriting abstract virtual methods that retrieve access control lists from an object and store them back. The component implements reading and writing **access allowed** , **access denied** and **system audit** information at both design and run time. One of the most exciting features of component is its ability to work with access denied entries. Note that Windows NT 4.0 standard tools do not support editing of access denied entries. At the moment NTSet library has three descendant of TNTAbstractSecurity component. They apply to the following objects:

TNTShare	shared devices
TNTFileSecurity	files and folders
TNTRegSecurity	registry keys

TObject
|
TPersistent
|
TComponent
|
TNTPersistent
|
TNTAbstractSecurity

TNTAbstractSecurity methods

TNTAbstractSecurity

TakeOwnerShip

TNTAbstractSecurity properties

TNTAbstractSecurity

In TNTAbstractSecurity

ObjectOwner
ControlAccess
AccessAllowed
AccessDenied
SystemAudit

Derived from TNTPersistent

MachineName

TNTAbstractSecurity.AccessAllowed

property AccessAllowed: TAccessList;

Description

This property returns pointer to the list of items giving access to the object. All the users which are not included in the AccessAllowed list will not have an access to the object. Each time you read this property component rereads actual information form the object. Read this property only to get pointer to the list or to refresh information. Use retrieved pointer for any other operations.

Note

If you do not have permissions to read file's security attributes you will get the "Access denied" exception. To understand the contradiction between AccessDenied and AccessAllowed lists see priority of access control.

TNTAbstractSecurity.AccessDenied

property AccessDenied: TAccessList;

Description

This property returns pointer to the list of items restricting access to the object. If a user or group is included into the AccessDenied list it somehow restricts their rights on the given file. If a user or group is not included in the AccessDenied list, their rights depend on the information from AccessAllowed list. Each time you read this property component rereads actual information form the disk. Read this property only to get pointer to the list or to refresh information. Use retrieved pointer for any other operations.

Note

If you do not have permissions to read file's security attributes you will get the "Access denied" exception. To understand contradiction between AccessDenied and AccessAllowed lists see priority of access control.

TNTAbstractSecurity.ControlAccess

property ControlAccess: boolean

Description

Property ControlAccess defines whether or not access to the object is controlled. If ControlAccess has "false" value then everyone has full access to the object. If ControlAccess is "true" and Access List is empty then no one has an access to the object.

Note

If you do not have permissions to read object's security attributes you will get the "Access denied" exception.

TNTAbstractSecurity.ObjectOwner

TNTAbstractSecurity

property ObjectOwner: string;

Property returns string specifying the account of user who has ownership over the object. Be careful when writing this property: security requirements do not allow to assign the ownership to the another user. The only name which is allowed to assign to this property is the name of currently logged on user. The better way to take ownership is to use TakeOwnerShip method;

TNTAbstractSecurity.SystemAudit

TNTAbstractSecurity Example

property SystemAudit: TAccessList;

This property returns pointer to the list of items specifying access audit for the object. The system maintains the "Security" section in the event log and writes in this section when specified event occurs. Remember that it is not enough to fill SystemAudit list, security audit to be in action. Check audit policy settings using Windows NT User Manager. Each time you read this property component rereads actual information form the disk. Read this property only to get pointer to the list or to refresh information. Use retrieved pointer for any other operations.

Note
If you do not have permissions to read file's security attributes you will get the "Access denied" exception.

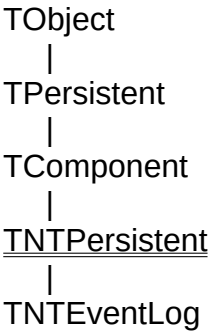
TNTAbstractSecurity.TakeOwnership

TNTAbstractSecurity

procedure TakeOwnership;

Description

Use this procedure instead of assigning owner of the object. Windows NT security system does not allow to assign any user as an owner of object. Only the name or currently logged on user is the valid choice. The procedure automatically defines the current user and tries to assign it as an object's owner. If the procedure fails it raises an exception.


TObject
TPersistent
TComponent
TNTPersistent
TNTEventLog

TNTEventLog component

Hierarchy Properties Methods Events

Unit
EventLog

TNTEventLog component is developed to allow to work with Windows NT event log as simple as with TStringList. TNTEventLog component encapsulates the set of relevant functions which are described in Windows API. Though you still can use API functions you have no need to do it.

The project Eventer.dpr demonstrates the main features of TNTEventLog component. Writer.dpr demonstrates technique of writing events into event log.

TNTEventLog events

TNTEventLog

OnChange

TNTEventLog methods

In TNTEventLog

Add
AddObject
BackupEventLog
Clear
GetEventSources

Derived from TNTPersistent

GetAccounts
GetGlobalGroups
GetLocalGroups
GetPrimaryDomainServerName
GetServers
GetUsers

TNTEventLog properties

TNTEventLog Legend

In TNTEventLog

- Active
- BackupFileName
- Count
- IncludeUserName
- Items
- Handle
- Objects
- SourceName

WriteObject

Derived from TNTPersistent

- LocalComputer
- LogonAs
- MachineName

TNTEventLog.Active

TNTEventLog Example

property Active: boolean;

Use Active property to establish / break a connection to an event log on the specified computer.

TNTEventLog.Add

TNTEventLog See also Example

procedure Add(EventType: TEventType; S: string);

Description

Use procedure Add to add string message without additional data into event log.
Call this procedure only if Active property is true and you've opened flesh event log (not backup file).New message will be added at the end of opened event log. If you want to use the whole set of capabilities of event log - use the method AddObject

TNTEventLog.AddObject

procedure AddObject(AObject: TEventItem);

Description

Use procedure AddObject to add a new message into event log.
Call this procedure only if Active property is true and you've opened flesh event log (not backup file). The new message will be added at the end of opened event log.

TNTEventLog.BackupEventLog

procedure BackupEventLog;

Description

BackupEventLog creates backup file of currently opened section of event log.
You should set BackupFileName property before calling this method.
If specified file already exists than procedure fails.

TNTEventLog.BackupFileName

TNTEventLog Example

property BackupFileName: string

Description

Use this property to specify the name of the file you are going to save event log in. This property must be set before call BackupEventLog procedure. The another reason to set BackupFileName property is to open backup file of event log.
Set BackupFileName before setting Active in order to open backup file of event log.

TNTEventLog.Clear

TNTEventLog Example

Clear deletes all the items from opened event log.

procedure Clear;

Description

Call clear to empty the list of events. The name of section to be erased is defined by SourceName property. Do not call this procedure if you have opened backup file of event log.

TNTEventLog.Count

TNTEventLog Example

property Count: integer; **read only**

This property reports number of items in the opened event log. Use it only if Active property is true.

TNTEventLog.Handle

property Handle: THandle; **read only**

This property contains the handle of event log. It is valid only when Active property is true.
You may use Handle property to call windows API functions directly.

TNTEventLog.IncludeUserName

property IncludeUserName: boolean;

Description

The Value of this property defines if Name of currently connected user is included into the message written into event log.
Set this property before call Add and AddObject procedures.

TNTEventLog.Items

property Items[Index: integer]: string; **read only**;

Description
Use items to read description of event at particular position. This property returns formatted string.
If source for the message is not found in the registry or registry has wrong information then
Items returns message:

"The description for Event ID (%d) in Source (%s) could not be found. It contains the following insertion string(s): "
Strings of message are following.

TNTEventLog.Objects

TNTEventLog Example

property Objects[Index: integer]: TEventItem;

Use *Objects* to retrieve full information about the particular event.
Objects returns pointer to object. Do not destroy this object. It will be destroyed automatically.

TNTEventLog.OnChange

TNTEventLog Example

TNotifyEvent = **procedure** (Sender: TObject) **of object**;

property OnChange: TNotifyEvent

Description
The OnChange event occurs when there is a change in opened event log.

TNTEventLog.SourceName

property SourceName: string;

Description

Contains string that specifies the name of the source to be opened. The source name must be a subkey of a logfile entry under the EventLog key in the registry. For example, the source name TheApp would be valid if the registry had the following form:

```
HKEY_LOCAL_MACHINE
  System
    CurrentControlSet
      Services
        EventLog
          Application
           TheApp
          Security
          System
```

If the source name cannot be found, the event logging service uses the Application logfile with no message files for the event identifier or category.

TNTFileSecurity component

Hierarchy Properties Methods

Unit
FilSecur

Usage
NTFS only

TNTFileSecurity component gives an access to the Windows NT sanctuary: the security system. Now Delphi programmers have powerful component which allows to watch over file's security at both design and run time. The security attributes can be assigned only to the files or folders which lie on NTFS partition. Having set FileName determine a type of the partition using FileSystem property. Use AccessAllowed to allow an access to the file or directory for the specified users or AccessDenied property to restrict an access. Use SystemAudit property to examine attempts of an (un)authorized access. Remember that assigning security attributes to files differs from that to folders.

SFiler.dpr project demonstrates the main features of TNTFileSecurity component.

Note:
To use the whole set of TNTFileSecurity's properties and methods you must have administrator permissions.

TNTFileSecurity methods

TNTFileSecurity

Derived from TNTAbstractSecurity

TakeOwnerShip

Derived from TNTPersistent

GetAccounts
GetGlobalGroups
GetLocalGroups
GetPrimaryDomainServerName
GetServers
GetUsers

TNTFileSecurity properties

TNTFileSecurity Legend

In TNTFileSecurity

- DriveType
- FileName
- FileOwner
- FileSystem

Derived from TNTAbstractSecurity

- AccessAllowed
- AccessDenied
- ControlAccess
- SystemAudit

Derived from TNTPersistent

- LocalComputer
- LogonAs
- MachineName

TNTFileSecurity.FileName

property FileName: string;

Description

FileName defines the name of the file or directory to work with. Make sure that you have written proper value into this property before undertaking any further steps.

TNTFileSecurity.FileOwner

TNTFileSecurity

property FileOwner: string;

Property returns string specifying the account of user who has ownership over the file or directory. Be careful when writing this property: security requirements do not allow to assign the ownership to the another user. The only name which is allowed to assign to this property is the name of currently logged on user. The better way to take ownership is to use TakeOwnerShip method;

TNTFileSecurity.FileSystem

property FileSystem: string; **read only**;

Description

Not all file systems have ability to store security attributes. Make sure that the file system of the drive, on which the file lies, supports security. FileSystem is a string representation of file system's type.

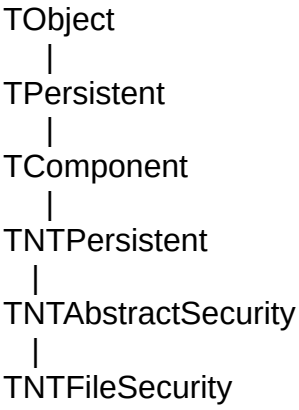
TNTFileSecutity.DriveType

TNTFileSecurity

property DriveType: TDriveType; **read only**

Description

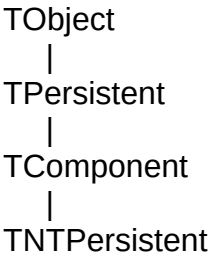
Property returns the type of the drive on which the file or directory FileName is.


TObject
TPersistent
TComponent
TNTPersistent
TNTAbstractSecurity
TNTFileSecurity

TNTPersistent component

Hierarchy Properties Methods

This component is an abstract ancestor for other NT Set's components. It has a number of common properties and method for its descendants. There is no need to create instances of this component.


TObject
TPersistent
TComponent
TNTPersistent

TNTPersistent methods

TNTPersistent

- GetAccounts
- GetDomains
- GetGlobalGroups
- GetLocalGroups
- GetPrimaryDomainServerName
- GetServerNameForDomain
- GetServers
- GetUsers

TNTPersistent properties

TNTPersistent

public

LocalComputer
LogonAs
MachineName

TNTPersistent.GetAccounts

TNTPersistent

function GetAccounts(AServerName: string; AFilter: DWORD): TStringList;

Description

The function returns list of accounts from the server of destination. It creates the object of type TStringList and fills it with data. You must destroy the list after you no longer need it.

Parameters

AServerName

The string identifies the computer of destination from which account information to be returned. The empty string specifies the local computer.

AFilter

Specifies kind of accounts to retrieve. The following values and their combinations are possible:

Value	Meaning
ACCOUNT_LOCAL_GROUP	retrieves local groups
ACCOUNT_GLOBAL_GROUP	retrieves global groups (if any)
ACCOUNT_USER	retrieves users
ACCOUNT_WELL_KNOWN	retrieves well known users
ACCOUNT_ALL	retrieves all accounts

TNTPersistent.GetGlobalGroups

TNTPersistent

function GetGlobalGroups(AServerName: string): TStringList;

Description

The function returns list of global groups from the server of destination. It creates the object of TStringList type and fills it with data. You must destroy the list after you no longer need it. *AServerName* string identifies the destination computer. An empty value specifies the local one.

TNTPersistent.GetLocalGroups

TNTPersistent

function GetLocalGroups(AServerName: string): TStringList;

Description

The function returns list of local groups from the server of destination. It creates the object of TStringList type and fills it with data. You must destroy the list after you no longer need it. *AServerName* string identifies the destination computer. An empty value specifies the local one.

TNTPersistent.GetPrimaryDomainServerName

TNTPersistent

function GetPrimaryDomainServerName: string;

Description

Returns server name of primary domain controller (if any).

TNTPersistent.GetServerNameForDomain

TNTPersistent

function GetServerNameForDomain(ADomainName: string): string;

Description

The function rerurns tha name of domain controller for the the domain specified by *ADomainName* parameter.

TNTPersistent.GetServers

TNTPersistent

```
function GetServers(AServerName: string): TStringList;
```

Description

Function creates and returns list of servers available on network. *AServerName* specifies computer to execute function on. If empty string is specified, function will be executed on the local computer. Remember to free retrieved TStringList after it is not necessary any more.

TNTPersistent.GetUsers

TNTPersistent

function GetUsers(AComputerName: string; Filter: TFilterAccountSet): TStringList;

Description

Function creates and returns list users' account registered on the server. Remember to free retrieved TStringList after you no longer need it.

Parameters

AComputerName identifies the server from which users' accounts to be obtained. An empty string specifies the local computer.
Filter specifies the kind of account(s) to retrieve.

TNTPersistent.LocalComputer

TNTPersistent

property LocalComputer: string; **read only;**

Description

Property returns string identifying the name of local computer.

TNTPersistent.LogonAs

TNTPersistent Example

property LogonAs: TLogonAs

Description

The property enables connection to the remote system using desirable logon. You may specify user name and password or you may leave the values empty. If empty values specified the component will not undertake any special actions when accessing remote system. If any values specified, the component will map inter-process communicator (IPC) of remote computer to the NULL local device. The mapping occurs at the moment of changing MachineName property. The component tries to delete the connection from the computer specified by the old value of MachineName property and creates connection to the computer identified by new MachineName's value.

TNTPersistent.MachineName

TNTPersistent

property MachineName: string;

Description

MachineName contains the name of the target computer. If empty string is specified, information will be obtained from the local computer. Note that when under Windows 95, you cannot leave this property empty. In this case it must contain valid universal name of a computer running Windows NT operation system. Most of the component's descendants override read and/or write methods of this property.

Note

In order to be able to control remote computer you must provide administrative account before assigning this property.

TNTPrivilege component

Hierarchy Properties Methods

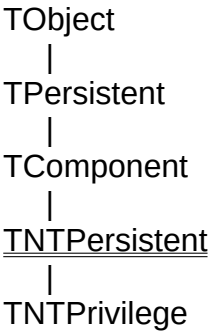
Unit
Privileg

Usage
Windows NT only

Microsoft Windows NT™ allows you to establish a full range of levels of security, from no security at all to the C2 level of security. However, the default configuration is highly relaxed. You may want to change these default settings to protect against accidental or deliberate changes to the way the computer is set up. TNTPrivilege component allows you to set desirable configuration of security privileges on a local or remote computer.

Set Privilege property and then apply Accounts to know what accounts have the given privilege. You may also use the opposite approach: set Account property and investigate the Privileges it has. You can manage remote computers as easy as the local one using MachineName property. In order to connect to the remote system under different account, use LogonAs property. The component can also adjust privilege list of the current process. Use the property Enabled to achieve this.

Project Rights.dpr demonstrates main features of TNTPrivilege.


TObject
TPersistent
TComponent
TNTPersistent
TNTPrivilege

TNTPrivilege properties

TNTPrivilege Legend

in TNTPrivilege

- Account
- Accounts
- DisplayName
- Enabled
- Privilege
- Privileges

Derived from TNTPersistent

- LocalComputer
- LogonAs
- MachineName

TNTPrivilege.Account

property Account: string;

Description

This property serves to specify the account you want to know privileges of. Privileges property returns account's list of privileges. You may receive all the account from the destination server with GetAccounts function.

TNTPrivilege.Accounts

property Accounts: TStrings;

Description

The property returns pointer to the list of accounts that have specified Privilege. Remember that each time when you apply the property the component rereads the list of accounts from server. Use this property to get the pointer to the list and use the received pointer to iterate through list.

TNTPrivilege.DisplayName

TNTPrivilege

property DisplayName: string; **read only**

Description

The DisplayName property retrieves a displayable name representing the privilege specified by Privilege. It returns the name in the local language for all privileges except for four rights:

SE_INTERACTIVE_LOGON_NAME
SE_NETWORK_LOGON_NAME
SE_BATCH_LOGON_NAME
SE_SERVICE_LOGON_NAME

The matter is that Windows NT privileges divided into two groups: 23 privileges and 4 rights. The system does not translate the names of rights.

TNTPrivilege.Enabled

property Enabled: boolean;

Description

This property enables or disables the Privilege for current process. Privilege property If you assign some privileges to the user, it means that any process, run by the user, acquires such a privilege. The privilege may be in two states: enabled and disabled. The Enabled property controls this.

If the component cannot change the privilege state it raises the exception "'Cannot change privilege state'". The possible reasons of error are:

- You try to change the state of one of four rights:
  - SE_INTERACTIVE_LOGON_NAME: Log on locally
  - SE_NETWORK_LOGON_NAME : Access this computer from network
  - SE_BATCH_LOGON_NAME : Log on as batch job
  - SE_SERVICE_LOGON_NAME: Log on as a service

  You cannot change the state of rights. You should add them to, or remove them from an account;
- The current user does not have the Privilege at all: you should add it using Accounts property;
- You just added the privilege t account and did not reboot the computer;

Note

This property affects only list of privileges of the current process on the local computer.

TNTPrivilege.GetPrivileges

TNTPrivilege

function GetPrivileges: TStringList;

Description

The function creates and returns string list with names of privileges defined in the system. There is no way to enumerate privileges, so the content of the list is hard coded. You must free the list after you no longer need it.

TNTPrivilege.Privilege

property Privilege: string;

Description

The property specifies the name of privileges you want to deal with. Having set this property you may find out its display name, retrieve list of Accounts that have this property, enable or disable the privilege for the current process.

TNTPrivilege methods

TNTPrivilege

in TNTPrivilege

GetPrivileges
GetPrivilegeDisplayName

derived from TNTPersistent

GetAccounts
GetGlobalGroups
GetLocalGroups
GetPrimaryDomainServerName
GetServers
GetUsers

TNTPrivilege.GetPrivilegeDisplayName

TNTPrivilege

function GetPrivilegeDisplayName(AValue: string): string;

Description

The function returns a displayable name representing the privilege specified by *AValue* parameter. The function returns string in local language. For more information see DisplayName.

TNTPrivilege.Privileges

property Privileges: TStrings;

Description

The property retrieves list of privileges for the specified Account on the computer identified by MachineName property. Every time when you apply the property, the component rereads actual information from the server. Use this property to receive pointer to the list. Deal with the pointer to enumerate the list.

TNTRegSecurity component

Hierarchy Properties Methods

Unit
RegSecur

TNTRegSecurity component is a descendent of TNTAbstractSecurity component that has been adjusted to deal with registry security. Now Delphi and C++Builder programmers have powerful component which allows to watch over registry's security at both design and run time. The component can read and write the registry key's discretionary and system access list as well as take ownership of registry key. Having set MachineName property, set the root key with RootKey property and registry path with CurrentPath. After it you can use AccessAllowed to allow an access to the registry key for the specified users or AccessDenied property to restrict an access. Use SystemAudit property to examine attempts of an (un)authorized access.

RegGuard.dpr project demonstrates the most exciting features of TNTRegSecurity component.

Note:
To use the whole set of TNTRegSecurity's properties and methods you must have administrator permissions. Otherwise you might get an exception "Access denied".

TNTRegSecurity methods

Derived from TNTAbstractSecurity

TakeOwnership

Derived from TNTPersistent

GetAccounts
GetGlobalGroups
GetLocalGroups
GetPrimaryDomainServerName
GetServers
GetUsers

TNTRegSecurity properties

TNTRegSecurity Legend

In TNTRegSecurity

- CurrentPath
- RootKey
- KeyOwner

Derived from TNTAbstractSecurity

- AccessAllowed
- AccessDenied
- ControlAccess
- SystemAudit

Derived from TNTPersistent

- LocalComputer
- LogonAs
- MachineName

TNTRegSecurity.CurrentPath


```
property CurrentPath: string;
```

Description

The property specifies the location of key in the registry of target machine. When setting desired value be sure that the path does not have the symbol "\" at the end, otherwise you may get an exception "2:Path not found". Set the property before reading or changing key's security attributes.

TNTRegSecurity.KeyOwner

property KeyOwner: string;

The property returns string specifying the account of user who has ownership of the registry key. Be careful when writing this property: security requirements do not allow to assign the ownership to the another user. The only name which is allowed to assign to this property is the name of currently logged on user. The better way to take ownership is to use TakeOwnerShip method;

TNTRegSecurity.RootKey

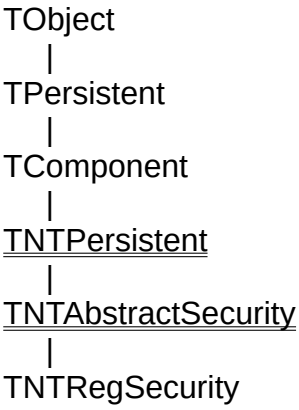
property RootKey: HKey; default HKEY_LOCAL_MACHINE

Description

The property specifies the root key of the target machine registry. The following values are defined in windows.pas unit and can be set as RootKey's values:

HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS
HKEY_CURRENT_CONFIG

The last value is valid only for local machine and will return an error "Invalid handle" if applied to remote server.


TObject
TPersistent
TComponent
TNTPersistent
TNTAbstractSecurity
TNTRegSecurity

TNTService

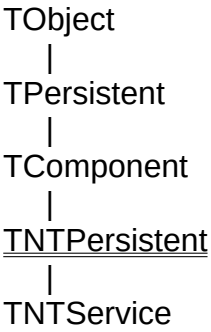
Hierarchy Properties Methods

Unit
TService

TNTService component is developed to add new services to Windows NT service control database as well as to start, stop, configure and delete them. It drastically simplifies tasks of controlling services. TNTService component encapsulates the set of relevant functions (except security functions) which are described in Windows API. Though you still can use API functions you have no need to do it.

In order to select a computer you are going to control services on, assign the name of the computer to MachineName property. Set ActiveManager property to establish connection to the service control manage on the computer of destination. Use GetServiceList method to enumerate services. If you want to know particular service's configuration, assign service name to ServiceName property, then set ActiveService.

The project SrvcMngr.dpr demonstrates the main features of TNTService component.


TObject
TPersistent
TComponent
TNTPersistent
TNTService

TNTService methods

TNTService

In TNTService

- Createservice
- ControlService
- DeleteService
- Startservice
- GetDependentServicesList
- GetOrderGroupList
- GetServiceList
- GetServiceDisplayName
- GetServiceKeyName
- QueryServiceStatus
- NotifyBootConfigStatus
- QueryServiceLockStatus

Derived from TNTPersistent

- GetAccounts
- GetGlobalGroups
- GetLocalGroups
- GetPrimaryDomainServerName
- GetServers
- GetUsers

TNTService properties

TNTService Legend

In TNTService

- ActiveManager
- ActiveService
- BinaryPathName
- DatabaseName
  - DBLocked
- DBLockHandle
- Dependencies
- DisplayName
- ErrorControl
- LoadOrder
- ManagerHandle
- ManagerAccess
- Password
- ServiceAccess
- ServiceHandle
- ServiceName
- ServiceStartName
- ServiceType
- StartType
- TagId

Derived from TNTPersistent

- LocalComputer
- LogonAs
- MachineName

TNTService.GetOrderGroupList method

TNTService

The function returns list of groups which define order of service's loading.

function GetOrderGroupList: TStrings;

Description

The function reads registry key HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\ServiceGroupOrder and retrieves list of groups. Each service may belong to zero or one group. Note that it is not always possible to read the registry on remote computer. If procedure fails, it raises an exception.

TNTShare component

Hierarchy Properties Methods

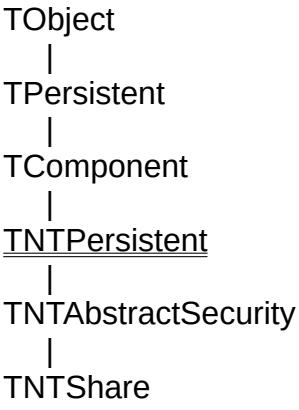
Unit
NTShare

TNTShare component encapsulates the set of Windows API functions which allow to configure shared devices on either local or network computers as well as retrieve information about users connected to given computer, configure network drivers and monitor resources used by other users

The project Shareman.dpr demonstrates the main features of TNTShare component.

Use property Resources to get a list of shared resources on local or remote computers. Remember to write the name of destination computer into the property MachineName before using any other properties. Resources' methods Add and Delete act like TList's methods and allow you to share and deshare resources. Sessions property retrieves variety of information about users and computers which established the sessions with computer of destination. Use Sessions' method Delete to close the session. It will also close all resources which are in use by that user or computer. Use Usages property to obtain a list of files, pipes and other resources which are in use at the moment. Connections property returns list of network drivers of local computer.

Note
Some of properties are available only if you have Administrator permissions on the computer of destination.


TObject
TPersistent
TComponent
TNTPersistent
TNTAbstractSecurity
TNTShare

TNTShare methods

TNTShare

Derived from TNTPersistent

GetAccounts
GetGlobalGroups
GetLocalGroups
GetPrimaryDomainServerName
GetServers
GetUsers

TNTShare properties

TNTShare Legend

In TNTShare

- Connections
- CurrentUsers
- MaxUsers
- Path
- Resources
- Sessions
- ShareComment
- ShareName
- ShareType
- Usages

Derived from TNTAbstractSecurity

- AccessAllowed
- AccessDenied

Derived from TNTPersistent

- LocalComputer
- LogonAs
- MachineName

TNTShare.CurrentUsers

property CurrentUsers: integer; **read only**;

Description
Property CurrentUsers returns the number of users using the shared resource. Set property MachineName to specify the destination computer before using this property. Remember to write the name of shared resource you are going to get information about into the property ShareName.

Note
You will get an exception "5: Access denied" if you do not have Administrator permissions on the destination computer.

TNTShare.MaxUsers

property MaxUsers: integer;

Description

MaxUsers Indicates the maximum number of concurrent connections that the shared resource can accommodate (unlimited if the specified value is "-1"). Set proper values into MachineName and ShareName properties before using MaxUsers.

Note

You will get an exception "5: Access denied" when either reading or writing this property if you do not have Administrator permissions on the destination computer.

TNTShare.Path

TNTShare Example

property Path: string; **read only**;

Description
Path contains the local path for the shared resource. For disks, *Path* is the path being shared. For print queues, *Path* is the name of the print queue being shared.

Note
You will get an exception "5: Access denied" if you do not have Administrator permissions on the destination computer.

TNTShare.ShareComment

TNTShare Example

property ShareName: string;

Description

ShareName is a string containing the share name of a resource, as remote users wiil see it . Remember to set MachineName and ShareName properties to before using this one.

TNTShare.ShareName

property ShareName: string;

Description

ShareName is a string containing the share name of a resource. This name will be visible for those browsing network. Set this property before retrieving any other information on resource. Uses Resources property to retrieve list of shared devices on the local or network computer. Note that ShareName property is automatically reset after MachineName property is changed.

TNTShare.ShareType

TNTShare

property ShareType: TShareType; **read only**;

Description

Specifies the type of network resource. Note that devices shared for administrative purposes (usually have '$'at the end of ShareName) have unknown share type.

TNTShare.Usages

property Usages: TUsageList

Description

This property returns pointer to the list of open files on MachineName. Each time you use Usages property component rereads the list of open files. Therefore use it only to obtain pointer to the list or to refresh information. Use retrieved pointer for any other operations.

TShareType type

TNTShare

TShareType = (stUnknown, stDisk, stPrint, stDevice, stIPC);

Specifies the type of network resource to connect to as well as the type of redirected device.

Value	Meaning
stUnknown	Unknown device
stPrint	Print queue
stDisk	Disk drive
stDevice	Communication device
stIPC	Interprocess communicator

TNTUserMan component

Hierarchy Properties Methods

Unit
Userman

TNTUserMan component is developed to to make a task of management of users under Windows NT™ as easy as possible. Now you have everything you need to add, delete, edit groups and users as well as to get variety of information about users on both local machine and network computers. Be careful when designing application using this component! It has full functionality at both design and run time.

UManager project demonstrates the main features of TNTUserMan component.

Note:
To use the whole set of TNTUserMan's properties and methods you must have administrator permissions.

TNTUserMan methods

TNTUserMan

In TNTUserMan

AddUser
ChangeUserName
DeleteGlobalGroup
DeleteLocalGroup
DeleteUser

Derived from TNTPersistent

GetAccounts
GetGlobalGroups
GetLocalGroups
GetPrimaryDomainServerName
GetServers
GetUsers

TNTUserMan properties

TNTUserMan Legend

In TNTUserMan

- GlobalGroupComment
- GlobalGroupId
- GlobalGroupMembers
- GlobalGroupName
- GlobalGroups
- LocalGroupComment
- LocalGroupMembers
- LocalGroupName
- LocalGroups
- MemberOfGlob
- MemberOfLocal
- UserInfo
- UserName
- Users

Derived from TNTPersistent

- LocalComputer
- LogonAs
- MachineName

TNTUserMan.ChangeUserName

TNTUserMan

procedure ChangeUserName(NewName: string);

Description

The procedure changes the name of the user specified UserName property into the new one which must be passed as parameter. If the procedure cannot change user name it raises an exception.

Note

Do not use UserName property to change the name of the user. This property only defines the user to retrieve information about.

TNTUserMan.DeleteGlobalGroup

TNTUserMan

procedure DeleteGlobalGroup(AName: string);

Description

The procedure removes the global group of users specified by *AName* parameter from the computer of destination. The same result may be achieved by using GlobalGroups property.

TNTUserMan.DeleteLocalGroup

TNTUserMan

procedure DeleteLocalGroup(AName: string);

Description

The procedure removes the local group of users specified by *AName* parameter from the computer of destination. The same result may be achieved by using LocalGroups property.

TNTUserMan.DeleteUser

procedure DeleteUser(AUserName: string);

Description
The procedure removes the account specified by *AUserName* parameter from the user database of destination computer. The same result may be achieved by using Users property.

TNTUserMan.GlobalGroupComment

property GlobalGroupComment: string;

Description
This property is used to get and set description of global group either on local machine or network computer.

Remark
GlobalGroupComment contains proper information only if GlobalGroupName is valid group name.

TNTUserMan.GlobalGroupId

property GlobalGroupId: integer; **read only;**

Description

This property returns the relative identifier of the account identified by GlobalGroupName property.

Remark

GlobalGroupComment contains proper information only if GlobalGroupName is valid group name.

TNTUserMan.GlobalGroupMembers

property GlobalGroupMembers: TStrings;

Description

GlobalGroupMembers property allows to retrieve and set list of members of particular global group on a server. Use this property to replace the whole list of members as well as to add(remove) particular user into(from) a global group See also MemberOfGlob property.

Note

Before using this property make sure that GlobalGroupName contains valid global group name.

TNTUserMan.GlobalGroupName

property GlobalGroupName: string

Description

This property contains the name of the global group of users on the selected computer. Set this property before retrieving any other information about global group (see also GlobalGroupComment, GlobalGroupMembers). You may get the list of all global groups on the given computer using GlobalGroups property.

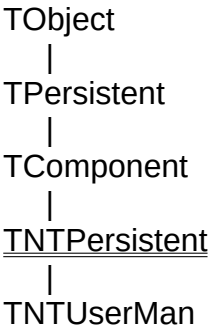
TNTUserMan.GlobalGroups

TNTUserMan Example

property GlobalGroups: TStrings;

Description

GlobalGroups property contains list of global groups on the server. Using this property you can retrieve list of global groups, add, delete group and replace the whole list of groups. Use GlobalGroups' Add and Delete methods to add or remove global group. Before using this property make sure that MachineName contains valid computer name.


TObject
TPersistent
TComponent
TNTPersistent
TNTUserMan

TNTUserman.AddUser

TNTUserMan

procedure AddUser(AUserName, APassword: string);

Description

The procedure creates new account in the destination computer's user database. You also may use Users property to create a new account. The parameter *AUserName* represents the name of account to be created. *APassword* is a password to use when logging in.

TOperatorFlags type

TNTUserMan TUserInfo

```
TOperatorFlag =(OP_PRINT, OP_COMM, OP_SERVER, OP_ACCOUNTS);
TOperatorFlags = set of TOperatorFlag;
```

Value	Meaning
OP_PRINT	The print operator privilege is enabled.
OP_COMM	The communications operator privilege is enabled.
OP_SERVER	The server operator privilege is enabled.
OP_ACCOUNTS	The accounts operator privilege is enabled.

TQueryServiceLockStatus type

TNTService See also

```
type
  TQueryServiceLockStatus = record
    fIsLocked: DWORD;
    lpLockOwner: PAnsiChar;
    dwLockDuration: DWORD;
  end;
```

Description

fIsLocked

Specifies whether the database is locked. If this member is nonzero, the database is locked. If it is zero, the database is unlocked.

lpLockOwner

Points to a null-terminated string containing the name of the user who acquired the lock.

dwLockDuration

Specifies the time, in seconds, since the lock was first acquired.

TResourceItem properties

TResourceItem TResourceList

- ShareName
- ShareType
- ShareComment
- MaxUsers
- CurrentUsers
- Path

TResourceItem type

Properties TResourceList TNTShare

TResourceItem represents an item in a TResourceList.

Description

A TResourceList holds a group of TResourceItem objects. All the properties of TResourceItem class are read only. To change some of resource's properties use TNTShare's properties such as ShareComment, MaxUsers and so on. Use Resources property and TResourceItem objects for enumeration tasks. TResourceItem objects are created and destroyed by TResourceList's Add, Delete and Clear methods. You will never need to create an instance of TResourceItem class explicitly. Use TResourceList's Items property to get pointers to TResourceItem instances maintained by TNTShare component.

TResourceItem.CurrentUsers

TResourceItem TResourceList

property CurrentUsers: integer; read only;

Description

CurrentUsers returns the number of users using the shared resource. This property is a replica of TNTShare.CurrentUsers used for enumeration purposes. Note that this property requires administrative permissions to retrieve right information but does not raise exception if you don't have them and returns "0".

TResourceItem.MaxUsers

TResourceItem TResourceList

property MaxUsers: integer; read only;

Description

MaxUsers indicates the maximum number of concurrent connections that the shared resource can accommodate. Unlimited number is identified by "-1". Note that you must have administrative permissions to get this information. Unlike to TNTShare.MaxUsers property this one does not create exception "Access denied" but returns "0".

TResourceItem.Path

TResourceItem TResourceList

property Path: string; read only;

Description

Path contains the local path for the shared resource. For disks, Path is the path being shared. For print queues, Path is the name of the print queue being shared. This property is just a replica of TNTShare. Path property used for iteration. Note that this property requires administrative permissions to retrieve proper information. If current user does have them, this property does not raise an exception but returns empty string.

TResourceItem.ShareComment

TResourceItem TNTShare

property ShareComment: string; read only;

Description

ShareComment contains an optional comment about the shared resource. This property is useful for retrieving information when iterating trough Resources list. Use TNTShare.ShareComment property to change the comment.

Example

```
var
  List: TResourceList;
begin
  List := Share1.Resources;
  for i := 0 to List.Count - 1 do
    if List[i].ShareComment = '' then
      ShowMessage('Resource '+List[i].ShareName + ' does not have comment yet!');
end;
```

TResourceItem.ShareName

TResourceItem TResourceList

property ShareName: string; read only;

ShareName is a string containing the share name of a resource. Use this property for iteration through Resources.

Example

```
var
  List: TResourceList;
begin
  List := Share1.Resources;
  for i := 0 to List.Count - 1 do
    begin
    Share1.ShareName := List[i].ShareName;
    Share1.ShareComment := 'Temporary opened resource';
    end;
end;
```

TResourceItem.ShareType

TResourceItem TResourceList

property ShareType: TShareType; read only;

Description

Specifies the type of network resource. Note that devices shared for administrative purposes (usually have '$'at the end of ShareName) have unknown share type. This property duplicates TNTShare.ShareType property and is used for enumeration tasks.

TResourceList methods

TResourceList TNTShare

- Add
- Clear
- Delete

TResourceList properties

TResourceList TNTShare

- Count
- Items

TResourceList type

Properties Methods TNTShare

TResourceList is a container for TResourceItem objects. It holds the shared devices of destination computer . The Count property contains the number of items in the list. Use the Add, Delete, Clear methods to add and delete connections. Usually you will not create instances of TResourceList class. Use TNTShare.Resources property instead to obtain a pointer to the list of shared devices maintained by TNTShare component.

Example

```
Share1.MachineName := '\\moon';
if Share1.Resources.Count = 0 then ShowMessage('No shared devices on server \
\moon');
```

TResourceList.Add

TNTShare TResourceList

Add shares a resource of server specified by MachineName property.

procedure Add(NetName, Path: string; ShareType: TShareType);

Description

Only members of the Administrators or Account Operators local group or those with Communication, Print, or Server operator group membership can successfully execute TResourceList.Add. The Print operator can add only Printer queues. The Communication operator can add only communication-device queues.

Example

```
Share1.Resources.Add('TEMP', 'c:\temp', stDisk);
Share1.Resources.Add('PaintJet on \\PC21', 'PaintJet', stPrint);
```

TResourceList.Clear

TResourceList TNTShare

procedure Clear;

Description

Clear deletes all items from a MachineName's list of shared resources, disconnecting all connections to the shared resources.

TResourceList.Count

TResourceList TNTShare

Count is the number of TResourceItem objects in the list.

property Count: Integer; read only;

Description

Read Count to determine the number of TResourceItem objects in the Items array.

Example

```
Share1.MachineName := '\\moon';
Label1.Caption := 'Server "\\moon" has : ' + IntToStr(Share1.Resources.Count) +'
shared devices' ;
```

TResourceList.Delete

TNTShare TResourceList

procedure Delete(Index: integer);

Description

Method deletes item from a MachineName's list of shared resources, disconnecting all connections to the shared resource. Only members of the Administrators or Account Operators local group or those with Communication, Print, or Server operator group membership can successfully execute NetShareDel. The Print operator can delete only Printer queues. The Communication operator can delete only communication-device queues.

Example

```
try
Share1.Resources.Delete(0);
except
ShowMessage('Cannot delete resource');
end;
```

TResourceList.Items

TResourceList TNTShare

Items is the array of object references.

property Items[Index: integer]: TResourceItem; **default;**

Description

Use Items to obtain a pointer to a specific TResourceItem object in the array. The Index parameter indicates the index of the object, where 0 is the index of the first object, 1 is the index of the second object, and so on. Use Items with the Count property to iterate through all of the objects in the list.

Example

```
var
  PList: TResourceList;
begin
  PList := Share1.Resources;
  for i := 0 to PList.Count - 1 do ListBox1.Items.Add(PList.Items[i].ShareName);
end;
```

TServiceAccess type

TNTService

type

TServiceAccess = set of TDesiredServiceAccess;
TDesiredServiceAccess = (S_ALL_ACCESS, S_CHANGE_CONFIG, S_ENUMERATE_DEPENDENTS, S_INTERROGATE, S_PAUSE_CONTINUE, S_QUERY_CONFIG, S_QUERY_STATUS, S_START, S_STOP, S_USER_DEFINED_CONTROL, S_DELETE);

Description

S_ALL_ACCESS
: Includes STANDARD_RIGHTS_REQUIRED in addition to all of the access types listed in this topic.

S_CHANGE_CONFIG
: Enables calling of the ChangeServiceConfig function to change the service configuration.

S_ENUMERATE_DEPENDENTS
: Enables calling of the EnumDependentServices function to enumerate all the services dependent on the service.

S_INTERROGATE
: Enables calling of the ControlService function to ask the service to report its status immediately.

S_PAUSE_CONTINUE
: Enables calling of the ControlService function to pause or continue the service.

S_QUERY_CONFIG
: Enables calling of the QueryServiceConfig function to query the service configuration.

S_QUERY_STATUS
: Enables calling of the QueryServiceStatus function to ask the service control manager about the status of the service.

S_START
: Enables calling of the Startservice function to start the service.

S_STOP
: Enables calling of the ControlService function to stop the service.

S_USER_DEFINE_CONTROL
: Enables calling of the ControlService function to specify a user-defined control code.

TServiceStates type

TNTService See also

type

```
TServiceState = (STATE_ACTIVE, STATE_INACTIVE);
TServiceStates = set of TServiceState;
```

Description

STATE_ACTIVE

Enumerates services that are in the following states: START_PENDING, STOP_PENDING, RUNNING, CONTINUE_PENDING, PAUSE_PENDING, and PAUSED.

STATE_INACTIVE

Enumerates services that are in the STOPPED state.

TServiceStatusClass type

TNTService See also

The TServiceStatusClass class contains information about a service.

```
type
  TServiceStatusClass = class
  public
    ServiceName:            string;
    DisplayedName:                  string;
    ServiceType:            TServiceTypes;
    CurrentState:           TCurrentState;
    ControlsAccepted:       TControlAcceptedSet;
    Win32ExitCode:          DWORD;
    ServiceSpecificExitCode:DWORD;
    CheckPoint:             DWORD;
    WaitHint::                      DWORD;
  end;
```

Description

dwWin32ExitCode

Specifies a Win32 error code that the service uses to report an error that occurs when it is starting or stopping. To return an error code specific to the service, the service must set this value to ERROR_SERVICE_SPECIFIC_ERROR to indicate that the dwServiceSpecificExitCode member contains the error code. The service should set this value to NO_ERROR when it is running and on normal termination.

dwServiceSpecificExitCode

Specifies a service specific error code that the service returns when an error occurs while the service is starting or stopping. This value is ignored unless the dwWin32ExitCode member is set to ERROR_SERVICE_SPECIFIC_ERROR.

dwCheckPoint

Specifies a value that the service increments periodically to report its progress during a lengthy start, stop, or continue operation. For example, the service should increment this value as it completes each step of its initialization when it is starting up. The user interface program that invoked the operation on the service uses this value to track the progress of the service during a lengthy operation. This value is not valid and should be zero when the service does not have a start, stop, or continue operation pending.

dwWaitHint

Specifies an estimate of the amount of time, in milliseconds, that the service expects a pending start, stop, or continue operation to take before the service makes its next call to the SetServiceStatus function with either an incremented dwCheckPoint value or a change in dwCurrentState. If the amount of time specified by dwWaitHint passes, and dwCheckPoint has not been incremented, or dwCurrentState has not changed, the service control manager or service control program can assume that an error has

occurred.

TServiceTypes type

TNTService

type

```
  TServiceTypes = set of TServiceType;
  TServiceType = (KERNEL_DRIVER, FILE_SYSTEM_DRIVER, ADAPTER,
RECOGNIZER_DRIVER, WIN32_OWN_PROCESS, WIN32_SHARE_PROCESS,
INTERACTIVE_PROCESS);
```

Description

WIN32_OWN_PROCESS
: A service-type flag that specifies a Win32 service that runs in its own process.

WIN32_SHARE_PROCESS
: A service-type flag that specifies a Win32 service that shares a process with other services.

KERNEL_DRIVER
: A service-type flag that specifies a Windows NT device driver.

FILE_SYSTEM_DRIVER
: A service-type flag that specifies a Windows NT file system driver.

INTERACTIVE_PROCESS
: A flag that enables a Win32 service process to interact

ADAPTER

RECOGNIZER_DRIVER

TSession properties

TSession TSessionList

- ClientName
- ClientType
- IdleTime
- OpenResourses
- SessionTime
- Transport
- UserFlags
- UserName

TSession type

Properties TSessionList

TSession represents an item in a TSessionList.

Description

A TSessionList holds a group of TSession objects. TSession objects are created by TSessionList class. You may close session using Clear and Delete methods. You will never need to create an instance of TSession class explicitly. Use TSessionList's Items property to get pointers to TSession instances maintained by TNTShare component.

TSession.ClientName

TSession TSessionList

property ClientName: string; read only;

Property contains the name of the computer that established the session.

TSession.ClientType

TSession TSessionList

property ClientType: string; read only;

Description

Specifies the type of client that established the session. this property returns empty string if you do not have membership in Administrators or Account Operators local groups.

TSession.IdleTime

TSession TSessionList

property IdleTime: TDateTime; read only;

This is the time for which session has been idle.

TSession.OpenResources

TSession TSessionList

property OpenResources: integer; read only;

Description

OpenResources returns the number of files, devices, and pipes opened during the session. Property returns "-1" if user does not have administrative permissions.

TSession.SessionTime

TSession TSessionList

property SessionTime: TDateTime; read only;

Specifies the time a session has been active.

TSession.Transport

TSession TSessionList

property Transport: string; read only;

Description

Specifies the name of the transport that the client is using to communicate with the server specified by MachineName. This property returns empty string if you do not have membership in Administrators or Account Operators local groups.

TSession.UserFlags

TSession TSessionList

property UserFlags: TSessionFlag; read only;

Describes how the user established the session. Only members of the Administrators or Account Operators local group can successfully retrieve this information. Otherwise sessUnknown is returned.

TSession.UserName

TSession TSessionList

property UserName: string; read only;

Property specifies the name of the user who established the session.

TSessionFlag type

TSession

Type describes session's characteristics.

TSessionFlag = (sessUnknown, sessNone, sessGuest, sessNoEncryption);

Value	Meaning
sessUnknown	Not enough permission to get information
sessGuest	Session is established using a guest account.
sessNoEncryption	Session is established without using password encryption.
sessNone	No one of previous flags.

TSessionList methods

TSessionList TNTShare

Delete

TSessionList properties

TSessionList TNTShare

Count

Items

TSessionList type

Properties Methods TNTShare

TSessionList is a container for TSession objects. It holds session list of server specified by MachineName property. The Count property contains the number of items in the list. Use the Delete and Clear methods to delete selected or all sessions connections. Usually you will not create instances of TSessionList class. Use TNTShare.Sessions property instead to obtain a pointer to the list of sessions maintained by TNTShare component.

Example

```
if Share1.Sessions.Count = 0 then ShowMessage('No connected users!')
```

TSessionList.Count

TSessionList TNTShare

Count is the number of TSession objects in the list.

property Count: Integer; read only;

Description

Read Count to determine the number of TSession objects in the Items array.

Example

```
Share1.MachineName := '\\moon';
Label1.Caption := 'Server "\\moon" has : ' + IntToStr(Share1.Sessions.Count) +'
established sessions' ;
```

TSessionList.Delete

TNTShare TSessionList

procedure Delete(Index: integer);

Description

Clear ends a session between MachineName server and a workstation. Note that Delete may end more than one session at once. If you are deleting session with empty UserName property all the sessions of ClientName computer will be terminated. It's recommended to reread Sessions property after call Delete method.

TSessionList.Items

TSessionList TNTShare

Items is the array of object references.

property Items[Index: integer]: TSession; default;

Description

Use Items to obtain a pointer to a specific TSession object in the array. The Index parameter indicates the index of the object, where 0 is the index of the first object, 1 is the index of the second object, and so on. Use Items with the Count property to iterate through all of the objects in the list.

Example

```
var
  PList: TSessionList;
begin
  PList := Share1.Sessions;
  for i := 0 to PList.Count - 1 do ListBox1.Items.Add(PList.Items[i].ClientName);
end;
```

TStartType

TNTService

type

TStartType = (BOOT_START, SYSTEM_START, AUTO_START, DEMAND_START, DISABLED);

Description

BOOT_START

Specifies a device driver started by the operating system loader. This value is valid only if the service type is KERNEL_DRIVER or FILE_SYSTEM_DRIVER.

SYSTEM_START

Specifies a device driver started by the IoInitSystem function. This value is valid only if the service type is KERNEL_DRIVER or FILE_SYSTEM_DRIVER.

AUTO_START

Specifies a device driver or Win32 service started by the service control manager automatically during system startup.

DEMAND_START

Specifies a device driver or Win32 service started by the service control manager when a process calls the Startservice function.

DISABLED

Specifies a device driver or Win32 service that can no longer be started.

TUsage properties

TUsage TUsageList

- ResourceId
- Permissions
- NumLocks
- PathName
- UserName

TUsage type

Properties TNTShare TUsageList

TUsage represents an item in a TUsageList.

Description

TUsageList holds a group of TUsage objects. TUsage objects are created by TNTShare component and destroyed by TUsageList's Delete and Clear methods. You will never need to create an instance of TUsage class explicitly. Use TUsageList's Items property to get pointers to TUsage instances maintained by TNTShare component.

TUsage.NumLocks

TUsage TUsageList

property NumLocks: integer; read only;

Description

NumLocks specifies the number of file locks on the file, device, or pipe.

TUsage.PathName

TUsage TUsageList

property PathName: string; read only;

Description

PathName property gives the path of the opened resource.

Example

```
var
  PUsages: TUsageList; i: integer;
begin
  PUsages := Share1.Usages;
  for i := 0 to PUsages.Count - 1 do ListBox1.Items.Add(PUsages[i].PathName);
end;
```

TUsage.Permissions

TUsage TUsageList

property Permissions: TAccessTypes;

Description

Specifies the access permissions of the opening application. This member can be any of the following values:

actRead	Permission to read data from a resource and, by default, to execute the resource.
actWrite	Permission to write data to the resource.
actCreate	Permission to create a resource; data can be written when creating the resource.

TUsage.ResourceId

TUsage TUsageList

property ResourceId: integer; read only;

Description

Specifies the identification number assigned to the resource when it is opened.

TUsage.UserName

TUsage TUsageList

property UserName: string; read only;

Description

String specifies which user (on servers that have user-level security) or which computer (on servers that have share-level security) opened the resource.

TUsageList methods

TUsageList TNTShare

Clear

Delete

TUsageList properties

TUsageList

Count

Items

TUsageList type

Properties Methods TNTShare

TUsageList is a container for TUsage objects. It holds list of open files on the server specified by MachineName property. The Count property contains the number of items in the list. Use the Delete and Clear methods to close selected or all files. Usually you will not create instances of TUsageList class. Use TNTShare.Usages property instead to obtain a pointer to the list of sessions maintained by TNTShare component.

Example

```
if Share1.Usages.Count = 0 then ShowMessage('No opened files!');
```

TUsageList.Clear

TUsageList TNTShare

Clear closes all resources opened on the server MachineName

procedure Clear; override;

Description

Call Clear to close all resources opened on the server MachineName. Call Delete to close particular one.

TUsageList.Count

TUsageList TNTShare

Count is the number of entries in the list.

property Count: Integer; read only;

Description

Read Count to determine the number of TUsage objects in the Items array.

Example

```
ShowMessage(IntToStr(Share1.Usages.Count)+ ' open resource(s) on the server
'+Share1.MachineName);
```

TUsageList.Delete

TUsageList TNTShare

procedure Delete(AIndex: integer);

Use Delete method to close resource.

Example

```
procedure Form1.btnClearClick(Sender: TObject)
var
  PUsages: TUsageList;
begin
  PUsages := Share1.Usages;
  while PUsages.Count > 0 do PUsages.Delete(0);
end;
```

TUsageList.Items

TUsageList TNTShare

Items is the array of object references.

property Items[Index: Integer]: TUsage; default;

Description

Use Items to obtain a pointer to a specific TUsage object in the array. The Index parameter indicates the index of the object, where 0 is the index of the first object, 1 is the index of the second object, and so on. Use Items with the Count property to iterate through all of the objects in the list.

TUserFlags type

TNTUserMan TUserInfo

type

TUserFlag = (F_SCRIPT, F_ACCOUNTDISABLE, F_HOMEDIR_REQUIRED, F_LOCKOUT, F_PASSWD_NOTREQD, F_PASSWD_CANT_CHANGE, F_DONT_EXPIRE_PASSWD);

TUserFlags = set of TUserFlag;

Value	Description
F_SCRIPT,	The logon script executed. This value must be set for LAN Manager 2.0 or Windows NT.
F_ACCOUNTDISABLE,	The user's account is disabled
F_HOMEDIR_REQUIRED,	The home directory is required. This value is ignored in Windows NT.
F_LOCKOUT,	The account is currently locked out. This value cannot be used to lock a previously locked account.
F_PASSWD_NOTREQD,	No password is required
F_PASSWD_CANT_CHANGE,	The user cannot change the password.
F_DONT_EXPIRE_PASSWD	Represents the password, which should never expire on the account. This value is valid only for Windows NT.

TUserInfo class

TNTUserMan

Type

```
  TUserInfo      = class(TPersistent)
  public
    property UserSID
  published
    property AccountExpires
    property BadPasswordCount
    property CodePage
    property Comment
    property CountryCode
    property Domain
    property FullName
    property HomeDir
    property HomeDirDrive
    property LastLogOff
    property LastLogon
    property LogonCount
    property LogonHours
    property LogonServer
    property MaxStorage
    property OperatorRights
    property Options
    property Password
    property PasswordDate
    property PasswordExpired
    property Privilege
    property Profile
    property ScriptPath
    property Workstations
  end;
```

TUserInfo.HomeDirDrive

property HomeDirDrive: string;

Description

Specifies the drive letter assigned to the user's home directory for logon purposes.

TUserInfo.LogonHours

property LogonHours: TLogonHours

Description

The property specifies the time during which user can log on to the computer of destination. The property defines weekly time table with one hour precision. You can specify enabled or disabled state of connection for each of 168 hours of week.

TUserInfo.PasswordExpired

property PasswordExpired: boolean;

Description

Determines whether the password of the user has expired. It returns zero if the password has not expired (and nonzero if it has). Specify "true" to indicate that the user must change password at next logon. Note that you cannot specify "false" to negate the expiration of a password that has already expired.

TUserInfo.Profile

property Profile: string;

Description

Specifies a path to the user's profile. This value can be an empty string, a local absolute path, or a UNC path.

TUserPriv type

TNTUserMan TUserInfo

type

TUserPriv = (USR_PRIV_UNKNOWN, USR_PRIV_GUEST, USR_PRIV_USER, USR_PRIV_ADMIN);

These values specify the level of privilege assigned to the user UserName

TNTService.TagId

TNTService

property TagId: integer;

Description

32-bit variable that receives a unique tag value for this service in the group specified in the LoadOrder property parameter. If no tag is requested, this parameter can be 0. You can use a tag for ordering service startup in a load ordering group by specifying a tag order vector in the registry located at:
HKEY_LOCAL_MACHINE\System\ CurrentControlSet\Control\GroupOrderList.
Tags are only evaluated for KERNEL_DRIVER and FILE_SYSTEM_DRIVER type services that have BOOT_START or SYSTEM_START start types.

TNTUserMan.UserInfo

property UserInfo: TUserInfo;

Description

This property is a class which allows to get variety of information about particular user. use this property having set proper values into MachineName and UserName properties.

TNTUserMan.UserName

property UserName: string;

Description

UserName contains the name of user registered on the selected computer. Set this property before retrieving any information about user. You can get the whole list of users registered on the given computer using Users property.

TUserInfo.UserSID

property UserSID: PSID;

Description

This property is a pointer to SID structure. SID stands for Security Identifier. It uniquely identifies each user. You can use this property for direct calls of API functions. This property has valid value only when UserName property contains valid user's name.

TNTUserMan.Users

property Users: TStrings;

Description

Users property contains list of users registered on the selected computer. Use Users' Add and Delete
methods do to add and remove users. Make sure that MachineName has a valid computer name before using this property.

Remark

New user is created with a password and comment coinciding with UserName.

Windows NT vs. Windows 95

What is in the name? "Component Set for Windows NT" has been developed to give computer programmers an ability to use low level Windows NT features not diving into API. But, it does not mean that this collection of component cannot work under Windows 95. It can. In this case it may be used to administrate Windows NT servers and workstations remotely. You will see a special note if some property or method does not work under Windows 95. It usually applies to those of them which affect local machine.

Standard Windows 95 setup does not install libraries for remote Windows NT administration. You should install them separately. The list of necessary files below may be found on Windows NT server CD-ROM:

RADMIN32.DLL
ACLEDIT.DLL
FPNWCLNT.DLL
NETMSG.DLL
NETUI0.DLL
NETUI1.DLL
NETUI2.DLL
NTLANUI.DLL
PRTQ32.DLL
RLOCAL32.DLL
RSHX32.DLL

The path to these libraries must be declared in system PATH variable. Each component of NTSet collection automatically detects version of operation system. They import API functions from RADMIN32.DLL when under Under Windows 95 and from NETAPI32.DLL, ADVAPI32.DLL when under Windows NT.

TUserInfo.Workstations

property Workstations: string;

Description

Property contains the names of workstations from which the user can log on. As many as eight workstations can be specified; the names must be separated by commas (,). An empty string indicates that there is no restriction. To disable logons from all workstations to this account, set the F_ACCOUNTDISABLE value in the Options.

TEventItem properties

TEventItem Legend

- Computername
- Data
- DataBytes
- DataLength
  - EventCategory
  - EventID
  - EventType
- Number
- SourceName
- Strings
- TimeGenerated
- TimeWritten
- UserName

TEventItem.ComputerName

TEventItem

property Computername: string **read only**;

Description

The property returns a string specifying the name of the computer that generated this event.

TEventItem.Data

TEventItem

property Data: Pointer **read only**;

Description

The property returns pointer to the block of memory with data, accompanying the event. This information could be something specific (a disk driver might log the number of retries, for example), followed by binary information specific to the event being logged and to the source that generated the entry. If there is no data, the property returns *null*.

Note

This property does not exist in ActiveX version of component library. Use property DataBytes that returns the same information in a way, convinient for VB programmers.

TEventItem.DataBytes

TEventItem

property DataBytes: TByteList; **read only**

Description

The property represents the array of bytes that are contained within the event record. The property returns the same information as Data property, but may be accessed through ActiveX interface. The Data property cannot be used in ActiveX version.

TByteList type

TEventItem

```
   TByteList = class
   public
      property Item[AIndex: integer]: byte;        // returns one byte from the array
      property Count: integer;                     // returns number of bytes in the array
      procedure Clear;                             // clears the array of byte, resets count
to zero
      procedure Add(AValue: byte);                 // adds new byte into array
  end;
```

TEventItem.DataLength

TEventItem

property DataLength: DWORD; **read only**;

Description

The property returns number of byte in the block of memory, pointed by Data property.

Note

This property is not reachable in ActiveX version of component library. Use property DataBytes.Count that returns the same information in a way, convinient for VB programmers.

TEventItem.EventCategory

TEventItem

property EventCategory: WORD;

Description

Specifies a subcategory for this event. This subcategory is source specific.

TEventItem.EventID

TEventItem

property EventID: DWORD;

Description

Identifies the event. This is specific to the source that generated the event log entry, and is used, together with SourceName, to identify a message in a message file that is presented to the user while viewing the log.

TEventItem.EventType

TEventItem

property EventType: TEventType;

Description

Specifies the type of event. The property can have one of the following values:

Value	Meaning
EVT_SUCCESS	Success event
EVT_ERROR	Error event
EVT_WARNING	Warning event
EVT_INFORMATION	Information event
EVT_AUDIT_SUCCESS	Success Audit event
EVT_AUDIT_FAILURE	Failure Audit event

TEventItem.Number

TEventItem

property Number: integer; **read only**

Description

The property returns sequence number of the event in the event log.

TEventItem.SourceName

TEventItem

property SourceName: string; **read only**

Description

Returns string specifying the name of the source (application, service, driver, subsystem) that generated the entry. This is the name used to retrieve from the registry the name of the file containing the message strings for this source. It is used, together with the event identifier, to get the message string that describes this event.

TEventItem.Strings

TEventItem

property Strings: TStrings; **read only**

Description

The property holds insertion strings to be used in creating a message, when a user views an event log.

TEventItem.TimeGenerated

TEventItem

property TimeGenerated: TDateTime; **read only**

Description

The time at which this entry was submitted.

TEventItem.TimeWritten

TEventItem

property TimeWritten: TDateTime **read oinly**;

Description

Specifies the time at which this entry was received by the service to be written to the logfile.

TEventItem.UserName

TEventItem

property UserName: string; **read only**;

Description

Specifies the name of active user at the time this event was logged. This property may be empty.

TEventType methods

TEventType

SetData

TEventType.SetData

TEventType

procedure SetData(AData: Pointer; DataSize: integer);

Description

The procedure writes the block of data into inner structures of TEventType class. The data set may be retieved through Data property. You may need to use this property when adding new record into event log with AddObject method.

Note

This procedure does not exist in ActiveX version of component library. Use property DataBytes.Add that sets the data in a way, convinient for VB programmers.

TNTEventLog.GetEventSources

TNTEventLog

function GetEventSources: TStrings;

Description

The function returns the list of strings filled with the names of event sources. The procedure retrievs the list from the registry.

TNTEventLog.WriteObject

TNTEventLog

property WriteObject: TEventItem;

Description

This property is a pre-created object of TEventItem type. It may be used with AddObject method. It allows to avoid creation of a new object before calling the method and destroying it after.

TNTPersistent.GetDomains

NTPersistent

```
function GetDomains(AServerName: string): TStringList;
```

Description

Function creates and returns list of domains available on network. *AServerName* specifies computer to execute function on. If empty string is specified, function will be executed on the local computer. Remember to free retrieved TStringList after it is not necessary any more.