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

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.

Example

```
NTService.MachineName := '\\PC21';
NTService.ManagerAccess := [M_CONNECT, M_ENUMERATE_SERVICE];
NTService.ActiveManager := true;
```

TNTService.ActiveService

TNTService

property ActiveService: boolean;

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

Example

```
NTService.ActiveManager := true;
NTService.ServiceName     := 'Schedule';
NTService.ServiceAccess := [S_ALL_ACCESS];
NTService.ActiveService     := true;
```

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 turn to true affect the database.

```
NTService.ServiceName     := 'Schedule';
NTService.ServiceAccess := [S_ALL_ACCESS];
NTService.ErrorControl         =   ERROR_NORMAL;              // useless;
NTService.ActiveService     := true;
```

TuserMan.BadPasswordCount

TUserMan 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

TNTService

property BinaryPathName: string;

Description
Contains the fully qualified path to the service binary file.

Example

```
NTService.ActiveManager := true;
NTService.ServiceName := 'Schedule';
NTService.ActiveService := true;
Edit1.Text := NTService.BinaryPathName;
```

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.

TShare.Connections

TShare

property Connections: TConnectionList;

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.

Example

```
var
   i: integer;
   PConnections: TConnectionList;
begin
   PConnections := Share1.Connections;
   for i := 0 to PConnections.Count - 1 do
ListBox1.Items.Add(PConnections[i].LocalName);
end;
```

Contact information

TNTService, TEventLog, TUserMan, TShare, TFileSecurity are first in a row of components designed to use specific features of Windows NT™. This is the shareware version. Full version with source code is available for registered users.

E-mail contact@risq.belcaf.minsk.by
Visit us at http://risq.belcaf.minsk.by

Contact information

 TNTService component

 TEventLog component

 TUserMan component

 TShare component

 TFileSecurity component

TNTService.ControlService

TNTService

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

procedure ControlService(Code: TControlCode);

Example

```
  with NTService do
    begin
    ManagerAccess := [M_CONNECT, M_ENUMERATE_SERVICE];
    ServiceAccess := [S_ALL_ACCESS];
    ActiveManager := true;
    ServiceName     := 'NetLogon';
    ActiveService := true;
    ControlService(CONTROL_STOP);
    ActiveManager := false;
    end;
```

TUserInfo.CountryCode

property CountryCode: integer;

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

TNTService.CreateService method

TNTService

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.

Example

```
with NTService do
  begin
  ManagerAccess      := [M_CONNECT, M_CREATE_SERVICE];
  ServiceType        := [WIN32_OWN_PROCESS];
  ErrorControl       := ERROR_NORMAL;
  ServiceName        := 'MyService';
  DisplayName        := 'My Service'
  BinaryPathNAme     := 'c:\temp\myserv.exe';
  StartType := AUTO_START;
  ActiveManager := true;
  CreateService;
  ActiveManager := false;
  end;
```

LockHandle property

TNTService

property DBLockHandle: SC_LOCK; read only

property DBLockHandle contains lock to the specified service control manager database when DBLockHandle 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

TNTService

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.

Example

```
with NTService do
  begin
  ManagerAccess := NTService.ManagerAccess + [M_LOCK];
  ActiveManager := true;
  DBLocked := true;
  ActiveService := true;
  StartType := [DEMAND_START];
  ActiveService := false;
  DBLocked := false;
  ActiveManager := false;
  end;
```

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.

TNTService.DeleteService method

TNTService

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

Example

```
NTService.ServiceAccess := [S_ALL_ACCESS];
NTService.ServiceName     := 'MyService'
NTService.ActiveManager := true;
NTService.ActiveService := true;
NTService.DeleteService;
NTService.ActiveManager := false;
```

TNTService.Dependencies

TNTService

property Dependencies: TStrings;

Description

Contains the names of services or load ordering groups that must start before this service. doubly 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.

Example

```
NTService.Dependencies := Memo1.Lines;
```

TNTService.DisplayName

TNTService

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.

Example

```
NTService.ServiceName := 'Schedule';
NTService.ActiveService := true;
NTService.DisplayName   := 'TimeTable';
NTService.ActiveService := false;
```

UserInfo.Domain

property Domain: string read only

Description

Property contains name of 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

TEventLog

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 TEventLog 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.

TEventLog 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".

TUserInfo.FullName

property FullName: string;

Description

The property contains the full name of the user.

Example

```
   UserMan1.Users.Add('JS');
..UserMan1.UserName := 'JS';
..UserMan1.UserInfo.FullName := 'John Smith';
```

GetDependentServiceList method

TNTService

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;

Example

```
  NTService.ActiveManager := true;
  NTService.ServiceName     := 'LanmanWorkstation';
  NTService.ServiceAccess := [S_ALL_ACCESS];
  NTService.ActiveService := true;
  with NTService.GetDependentServiceList([STATE_ACTIVE, STATE_INACTIVE]) do
    try
    Caption := IntToStr(Count) + ' dependent services';
    finally
    Free;
    end;
  NTService.ActiveManager := false;
```

TUserMan.GetServers

TUserMan

function GetServers(AServerName: string): TStringList;

Description

The function lists all servers that are visible in the primary domain. AServerName is a string containing the name of the remote server on which the function is to execute. An empty string specifies the local computer.

TNTService.GetServiceDisplayName method

TNTService See also

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

Example

```
NTService.ActiveManager := true;
Caption := NTService.GetServiceDisplayName('PlugPlay');
NTService.ActiveManager := false;
```

GetServiceKeyName method

TNTService See also

The GetServiceKeyName 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.

function GetServiceKeyName(ADisplayName: string): string;

Note

ActiveManager property must be TRUE before this function called

Example

```
NTService.ActiveManager := true;
Caption := NTService.GetServiceKeyName('Plug and Play');
NTService.ActiveManager := false;
```

GetServiceList method

TNTService

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.

Example

```
NTService.ManagerAccess := [M_CONNECT, M_ENUMERATE_SERVICE];
NTService.ActiveManager := true;
with NTService.GetServiceList([STATE_ACTIVE], [PROCESS]) do
  try
  i := IndexOf('Spooler');
  if <> -1 then Caption := 'Spooler is active';
  finally
  Free;
  end;
```

TUserInfo.HomeDir

property HomeDir: string;

Description

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

TUserMan.LastLogOff

TUserMan TUserInfo

property LastLogOff: TDateTime; read only

Description

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

TUserMan.LastLogon

property LastLogon: TDateTime read only

Description

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

TNTService.LoadOrder

TNTService

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.

Example

```
NTService.LoadOrder          := 'SpoolerGroup';
```

TUserMan.LocalGroupComment

TUserMan

property LocalGroupComment: string;

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.

Example

```
   UserMan1.MachineName := '\\PC27';
..UserMan1.LocalGroupName := 'Domain guests';
..UserMan1.LocalGroupComment := 'All guests of domain BelCAF';
```

TUserMan.LocalGroupMembers

TUserMan

property LocalGroupMembers: TStrings;

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.

Example 1 - replacement of the list of members

```
var
  List : TStringList;
begin
  List := TStringList.Create;
  try
  List.Add('PC21\Sergei');
  List.Add('PC15\John');
  Userman1.LocalGroupMembers := List;
  finally
  List.Free;
  end;
end;
```

Example 2 - how to add a new member to the global group

```
  Userman1.LocalGroupMembers.Add('PC21\Sergei');
```

Example 3 - how to delete a member from the global group

```
with Userman1.LocalGroupMembers do  Delete(IndexOf('PC21\Sergei'));
```

TUserman.LocalGroupName

TUserMan

property LocalGroupName: string

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.

Example

```
  UserMan1.MachineName := '\\MOON';
..UserMan1.LocalGroupName := 'Power users';
```

TUserMan.LocalGroups

TUserMan

property LocalGroups: TStrings;

Description

TUserMan.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.

Example 1 - how to add a new local group

```
Userman1.LocalGroups.Add('My family');
```

Example 2 - how to delete a local group

```
with Userman1.LocalGroups do Delete(IndexOf('My family'));
```

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

TNTService

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.

Example

```
var
    List: TEnumList;
begin
  NTService.ManagerAccess := [M_CONNECT, M_ENUMERATE_SERVICE];
  NTService.ActiveManager := true;
  List := NTService.GetServiceList([STATE_ACTIVE, STATE_INACTIVE], [DRIVER,
PROCESS]);
  try
  ...
  finally
  List.Free;
  NTService.ActiveManager := false;
  end;
end;
```

TNTService.ManagerHandle

TNTService

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.

Example

```
var
  Buf: array[0..63] of char;
  BufLen: dword;
  b: boolean;
begin
  BufLen := SizeOf(Buf);
  NTService.ActiveManager := true;
  b := WinSVC.GetServiceDisplayName(NTService.ManagerHandle, 'Schedule',
Buf,BufLen);
  NTService.ActiveManager := false;
end;
```

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.

TUserMan.MemberOfGlob

TUserMan

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.

Example 1 - replacement of the list of membership

```
var
  List : TStringList;
begin
  List := TStringList.Create;
  try
  Userman1.MachineName := ''
  Userman1.UserName := 'Administrator'
  List.Add('Project INTRANET');
  List.Add('Oracle users');
  Userman1.MemberOfGlob := List;
  finally
  List.Free;
  end;
end;
```

Example 2 - how to add a membership in a global group

```
  Userman1.MemberOfGlob.Add('Informix users');
```

Example 3 - how to delete a membership in the global group

```
with Userman1.MemberOfGlob do   Delete(IndexOf('Informix users'));
```

MemberOfLocal

TUserMan

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.

Example 1 - replacement of the list of membership

```
var
  List : TStringList;
begin
  List := TStringList.Create;
  try
  Userman1.MachineName := ''
  Userman1.UserName := 'Administrator'
  List.Add('Power users');
  List.Add('Replicators');
  Userman1.MemberOfLocal := List;
  finally
  List.Free;
  end;
end;
```

Example 2 - how to add a membership in a local group

```
  Userman1.MemberOfLocal.Add('Power users');
```

Example 3 - how to delete a membership in the local group

```
with Userman1.MemberOfLocal do   Delete(IndexOf('Power users'));
```

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. .

Example

```
  UserMan1.Username := 'John Smith';
  UserMan1.UserInfo.Options := [F_SCRIPT, F_PASSWD_CANT_CHANGE,
F_DONT_EXPIRE_PASSWD];
```

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

TNTService

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.

Example

```
NTService.Password                := 'JohnSmith';
NTService.ServiceStartName        := 'PC21\Administrator';
```

TUserInfo.PasswordDate

property PasswordDate: TDateTime; read only

Description

The property represents the data when password was changed.

Priority of access control

TFileSecurity component has two lists which affect access rights on a given file. AccessAllowed grants access while AccessDenied revokes it. Generally, AccessDenied 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

TUserMan TUserInfo

property Privilege: TUserPriv read only

Description

Use this property to determine level of permissions of user.

Example

```
  Userman1.MachineName := '';
  Userman1.UserName := GetUserName(Buf, SizeOf(Buf));
  if UserMan1.UserInfo.Privilege <> USR_PRIV_ADMIN then
    ShowMessage('You must have ADMINISTRATOR permissions to install this
software')
```

QueryServiceLockStatus method

TNTService

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.

Example

```
  NTService.ManagerAccess := NTService.ManagerAccess +
[M_QUERY_LOCK_STATUS];
  NTService.ActiveManager := true;
  with NTService.QueryServiceLockStatus do
    if   fIsLocked > 0 then
       Memo1.Lines.Add(Format('User %S has been locking service database for %D
seconds', [lpLockOwner, dwLockDuration]));
  NTService.ActiveManager := false;
```

QueryServiceStatus method

TNTService

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

Example

```
NTService.ActiveManager := true;
NTService.ServiceName     := 'PlugPlay';
NTService.ServiceAccess := [S_ALL_ACCESS];
NTService.ActiveService := true;
with NTService.QueryServiceStatus do
  try
  btnStop.Enabled := ACCEPT_STOP in ControlsAccepted
  finally
  Free;
  end;
NTService.ActiveManager := false;
```

TShare.Resources

TShare

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.

Example

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

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.

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

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.

Example

```
NTService.ServiceAccess := [S_QUERY_CONFIG];
NTService.ActiveService := true;
```

TNTService.ServiceHandle

TNTService

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.

Example

```
var
  b: boolean;
begin
  NTService.ServiceAccess := [S_ALL_ACCESS];
  NTService.ServiceName    := 'Schedule';
  NTService.ActiveManager := true;
  NTService.ActiveService := true;
  b := WinSVC.DeleteService(NTService.ServiceHandle);
  NTService.ActiveManager := false;
end;
```

TNTService.ServiceName

TNTService

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.

Example

```
NTService.ServiceName       := EdtServiceName.Text;
NTService.ActiveManager     := true;
NTService.ActiveService     := true;
```

TNTService.ServiceStartName

TNTService

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.

Example

```
NTService.Password              := 'JohnSmith';
NTService.ServiceStartName      := 'Administrator';
```

TNTService.ServiceType

TNTService

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.

Example

```
NTService.ActiveService := true;
NTService.ServiceType := NTService.ServiceType+[INTERACTIVE_PROCESS];
NTService.ActiveService := false;
```

TShare.Sessions

TShare

property Sessions: TSessionList

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.

Example

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

TNTService.StartService method

TNTService

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.

Example

```
NTService.ServiceAccess := [S_ALL_ACCESS];
NTService.ServiceName     := 'MyService';
NTService.ActiveManager := true;
NTService.ActiveService := true;
NTService.StartService;
NTService.ActiveManager := false;
```

TNTService.StartType

TNTService

Specifies when to start the service.

property StartType: TStartType

Example:

```
NTService.StartType := DEMAND_START;
```

TAccessItem properties

- UserName
- Mask
- Flags

TAccessItem type

Properties TFileSecurity See also

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 FileName file or directory. 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 TFileSecurity

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 TFileSecurity

property Mask: TFileAccessMasks;

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.

Example

```
FileSecurity.DeniedList[0].Mask := [famRead, famExecute];
```

TAccessItem.UserName

TAccessItem TAccessList TFileSecurity

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 TFileSecurity

- Add
- Delete
- Clear

TAccessList properties

TAccessList TFileSecurity

- Items
- Count

TAccessList type

Properties Methods TFileSecurity

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 TFileSecurity component.

Example

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

TAccessList.Add

TAccessList TFileSecurity

Add adds new item to the end of the list

function Add(UserName: string; AMask: TFileAccessMasks; 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

```
FileSecurity1.AccessDenied.Add('Everyone',[famFullControl], [ ]);
```

TAccessList.Clear

TAccessList TFileSecurity

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 TFileSecurity

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 TFileSecurity

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 TFileSecurity

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 TShare

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 TFileSecurity

```
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 TShare

- LocalName
- RemoteName
- UserName

TConnection type

Properties TShare 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 TShare component.

TConnection.LocalName

TConnection TShare

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 TShare

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 TShare

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 TShare

- Add
- AddEx
- Clear
- Delete

TConnectionList properties

TConnectionList TShare

Items

Count

TConnectionList type

Properties Methods TShare 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 TShare.Connections property instead to obtain a pointer to the list of connections maintained by TShare component.

Example

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

TConnectionList.Add

TConnectionList TShare 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 TShare

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 TShare

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

TConnectionList TShare

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 TShare

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 TShare

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.

TDriveType type

TFileSecurity

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

TEventLog

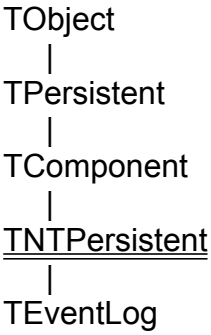
```
type
  TEventItem = class(TPersistent)
  public
    constructor Create;
    destructor   Destroy; override;
    procedure    Assign(Source : TPersistent); override;
    procedure    SetData(AData: Pointer; DataSize: integer);
    property TimeGenerated:      TDateTime;      read only;
    property TimeWritten:                TDateTime   read only;
    property EventID:                    DWORD;
    property EventType:                  TEventType;
    property Strings:                    TStringList;
    property UserName:                   string read only;
    property EventCategory:              WORD;
    property SourceName:                 string read only;
    property Computername:       string read only;
    property DataLength:                 DWORD read only;
    property Data:                       Pointer read only;
    property Number:                     integer read only
  end;
```

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.


TObject
TPersistent
TComponent
TNTPersistent
TEventLog

TEventLog component

Hierarchy Properties Methods Events

Usage

Windows NT only

TEventLog component is developed to allow to work with Windows NT event log as simple as with TStringList. TEventLog 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 TEventLog component. Writer.dpr demonstrates technique of writing events into event log.

TEventLog events

TEventLog

OnChange

TEventLog methods

Add
AddObject
BackupEventLog
Clear

TEventLog properties

TEventLog

Active
BackupFileName
Count
IncludeUserName
Items
Handle
MachineName
Objects
SourceName

TEventLog.Active property

TEventLog

property Active: boolean;

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

Example

```
EventLog.MachineName := '\\PC21';
EventLog.SourceName := 'Application';
EventLog.Active := true;
```

TEventLog.Add

TEventLog See also

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 Add at the end of opened event log. If you want to use the whole set of capabilities of event log - use procedure AddObject

Example

```
EventLog.SourceName := 'My Application';
EventLog.Active := true;
EventLog.Add(EVT_ERROR, 'No processor found');
EventLog.Active := false;
```

TEventLog.AddObject

TEventLog

procedure AddObject(AObject: TEventItem);

Description

Use procedure Add to add message 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 Add at the end of opened event log.

Example

```
var
  i: integer;
begin
 with TEventItem.Create do
  try
    EventID          := 100;
    EventType        := EVT_WARNING;
    Strings.Assign(Memo1.Lines);
    EventCategory  := 2;
    i                          := 1;
    SetData(@i, SizeOf(i));
    EventLog.AddObject(Obj);
  finally
  Free;
  end;
```

TEventLog.BackupEventLog

TEventLog

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.

Example

```
// Security section of event log will be saved.
   EventLog.SourceName := 'Security';
   EventLog.Active := true;
   EventLog.BackupFileName := 'c:\backup.evt';
   EventLog.BackupEventLog;
   EventLog.Active := false;
```

TEventLog.BackupFileName

TEventLog

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.

Example

```
  With TOpenDialog.Create(self) do
    try
    Filter       := 'Event Log files (*.EVT)|*.EVT|All files (*.*)|*.*';
    DefaultExt := 'EVT';
    if Execute then
      begin
      EventLog.BackupFileName := FileName;
      EventLog.Active := true;
      end;
    finally
    Free;
    end;
```

TEventLog.Clear

TEventLog

Clear deletes all the items from opened event log.

procedure Clear;

Description

Call clear to empty the list of events. Do not call this procedure if you have opened backup file of event log.

Example

The System section of event log will be erased.

```
   EventLog.SourceName := 'System';
..EventLog.backupFileName := '';
..EventLog.Active := true;
..EventLog.Clear;
..EventLog.Active := false;
```

TEventLog.Count

TEventLog

property Count: integer; read only

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

Example

```
EventLog.SourceName := 'Application';
EventLog.Active = true;
Edit1.Text := 'Number of items in Application section: '+IntToStr(EventLog.Count);
EventLog.Active = false;
```

TEventLog.Handle

TEventLog

property Handle: THandle; read only

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

Example

```
var OldestRecord:: DWORD;
begin
EventLog.Active := true;
GetOldestEventLogRecord(EventLog.Handle, OldestRecord);
end;
```

TEventLog.IncludeUserName

TEventLog

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.

Example

```
EventLog.Active := true;
EventLog.IncludeUserName := true;
EventLog.Add(EVT_INFORMATION, Memo1.Text);
EventLog.Active := false;
```

TEventLog.Items

TEventLog

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.

Example

```
Memo1.SetText(EventLog.Items[i]);
```

TEventLog.MachineName property

TEventLog

Property MachineName: string;

Description

MachineName contains the name of the target computer. If empty string is specified, connection will be established with the event log on the local computer.

Example

```
EventLog.MachineName := '\\PC21';
EventLog.SourceName := 'Application';
EventLog.Active := true;
```

TEventLog.Objects

TEventLog

property Objects[Index: integer]: TEventItem;

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

Example

```
S := EventLog.Items[0]+#$0D#$0A+' Message has '+
IntToStr(EventLog.Objects[0].DataLength) +' bytes of data';
```

TEventLog.OnChange

TEventLog

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

property OnChange: TNotifyEvent

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

Example

```
procedure TFrmLog.EventLogChange(Sender: TObject);
begin
   Beep;
   StatusBar1.SimpleText := 'New item in the Event log !';
end;
```

TEventLog.SourceName

TEventLog

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 MyApp would be valid if the registry had the following form:

```
HKEY_LOCAL_MACHINE
  System
    CurrentControlSet
      Services
        EventLog
          Application
            MyApp
          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.

Example

```
. EventLog.SourceName := 'Security';
  EventLog.Active := true;
  Edit1.Text := IntToStr(EventLog.Count)+' items in the security section'
```

TEventType

TEventLog

```
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

TFileAccessMasks type

TAccessItem TAccessList TFileSecurity

Type

TFileAccessMask = (famRead, famWrite, famExecute, famDelete, famPermissions, famOwnership, famFullControl);

TFileAccessMasks = set of TFileAccessMask;

Value	Meaning
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

TFileSecurity component

Hierarchy Properties Methods

Usage

Windows NT, NTFS only

TFileSecurity 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 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.

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

Note:

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

TFileSecurity methods

TFileSecurity

TakeOwnerShip

TFileSecurity properties

TFileSecurity

- AccessAllowed
- AccessDenied
- ControlAccess
- DriveType
- FileName
- FileOwner
- FileSystem
- SystemAudit

TFileSecurity.AccessAllowed

TFileSecurity

property AccessAllowed: TAccessList;

This property returns pointer to the list of items giving access to the file or directory specified by FileName property. All the users which are not included in the AccessAllowed list will not have an access to the file. 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 the contradiction between AccessDenied and AccessAllowed lists see priority of access control.

Example

```
var
  i: integer;
  List: TAccessList;
begin
   FileSecurity1.FileName := 'D:\Program Files\Projects';
   List := FileSecurity1.AccessAllowed;
   for i := 0 to List.Count - 1 do ListBox1.Items.Add(List[i].UserName);
end;
```

TFileSecurity.AccessDenied

TFileSecurity

property AccessDenied: TAccessList;

This property returns pointer to the list of items restricting access to the file or directory specified by FileName property. 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.

Example

```
var
  i: integer;
  List: TAccessList;
begin
   FileSecurity1.FileName := 'D:\Program Files\Projects';
   List := FileSecurity1.AccessDenied;
   for i := 0 to List.Count - 1 do ListBox1.Items.Add(List[i].UserName);
end;
```

TFileSecurity.ControlAccess

TFileSecurity

property ControlAccess: boolean

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

Note

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

Example

This example demonstrates how to prohibit an access to the file.

```
FileSecurity1.ControlAccess := false; // erase AccessAllowed and AccessDenied lists.
FileSecurity1.ControlAccess := true;
```

TFileSecurity.FileDrive

TFileSecurity

property FileDrive: string;

TFileSecurity.FileName

TFileSecurity

property FileName: string;

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.

Example

```
FileSecurity1.FileName := 'D:\temp';
if not (FileSecurity1.FileSystem) = 'NTFS' then Exit;
```

TFileSecurity.FileOwner

TFileSecurity

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;

TFileSecurity.FileSystem

TFileSecurity

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.

Example

```
if not (FileSecurity1.FileSystem) = 'NTFS' then Exit;
FileSecurity1.ControlAccess := true;
```

TFileSecurity.RemoteSystemName

TFileSecurity

TFileSecurity.SystemAudit

TFileSecurity

property SystemAudit: TAccessList;

This property returns pointer to the list of items specifying access audit for the file or directory denoted by FileName property. 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.

Example

```
var
  i: integer;
  List: TAccessList;
begin
   FileSecurity1.FileName := 'D:\Program Files\Projects';
   List := FileSecurity1.SystemAudit;
   for i := 0 to List.Count - 1 do ListBox1.Items.Add(List[i].UserName);
end;
```

TFileSecurity.TakeOwnerShip

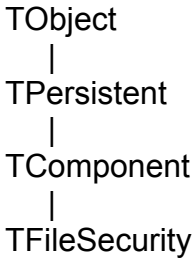
TFileSecurity

TFileSecutity.DriveType

TFileSecurity

public property DriveType: TDriveType; read only

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


TObject
TPersistent
TComponent
TFileSecurity

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.

TNTPersistent component

This component is an abstract ancestor for other NT Set's components. It has an important property MachineName which specifies the target computer for action of TNTService, TEventLog, TUserMan, TShare components. It also has a public function GetServers which returns list of available servers on network. There is no need to create instances of this component.

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 got TStringList after it is not necessary any more.

TNTService

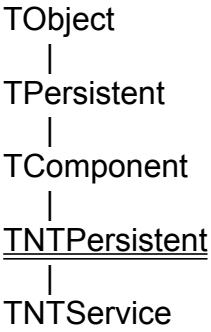
Hierarchy Properties Methods

Usage

Windows NT only

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.

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


TObject
TPersistent
TComponent
TNTPersistent
TNTService

TNTService methods

TNTService

- CreateService
- ControlService
- DeleteService
- StartService
- GetDependentServicesList
- GetServiceList
- GetServiceDisplayName
- GetServiceKeyName
- QueryServiceStatus
- NotifyBootConfigStatus
- QueryServiceLockStatus

TNTService properties

TNTService

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

TNTService.MachineName

TNTService

Property MachineName: string;

Description

MachineName contains the name of the target computer. If empty string is specified, connection will be established with the service control manager on the local computer.

Example

```
NTService1.MachineName       := '\\PC21'
NTService1.ManagerAccess     := [M_CONNECT];
NTService1.ActiveManager     := true;
```

TOperatorFlags type

TUserMan 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 TShare

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 TShare'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 TShare 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 TShare.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 TShare.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 TShare. 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 TShare

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 TShare.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 TShare.ShareType property and is used for enumeration tasks.

TResourceList methods

TResourceList TShare

- Add
- Clear
- Delete

TResourceList properties

TResourceList TShare

- Count
- Items

TResourceList type

Properties Methods TShare

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 TShare.Resources property instead to obtain a pointer to the list of shared devices maintained by TShare component.

Example

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

TResourceList.Add

TShare 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 TShare

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 TShare

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

TShare 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 TShare

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);

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

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 TShare 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 TShare

Delete

TSessionList properties

Count
Items

TSessionList type

Properties Methods TShare

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 TShare.Sessions property instead to obtain a pointer to the list of sessions maintained by TShare component.

Example

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

TSessionList.Count

TSessionList TShare

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

TShare 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 TShare

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;
```

TShare component

Hierarchy Properties

TShare 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 TShare 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
|
TShare

TShare properties

TShare

Connections

CurrentUsers

MachineName

MaxUsers

Path

Resources

Sessions

ShareComment

ShareName

ShareType

Usages

TShare.CurrentUsers

TShare

property CurrentUsers: integer; read only;

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.

Example

```
Share1.MachineName := '\\PC21';
ShareName = 'Database';
try
Edit1.Text := IntToStr(Share1.CurrentUsers);
except
Edit1.Text := 'Access denied';
end;
```

TShare.MachineName

TShare

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.

Example

```
Share1.MachineName    := '\\PC21'
```

TShare.MaxUsers

TShare

property MaxUsers: integer;

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.

Example

```
Share1.MachineName := '\\PC21';
ShareName = 'Database';
Share1.MaxUsers := 5;
```

TShare.Path

TShare

property Path: string; read only;

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.

Example

```
Share1.MachineName := '\\PC21';
ShareName = 'Database';
try
Edit1.Text := Share1.Path;
except
Edit1.Text := 'Access denied';
end;
```

TShare.ShareComment

TShare

property ShareName: string;

Description

ShareName is a string containing the share name of a resource. Remember to set MachineName and ShareName properties to before using this one.

Example

```
Share1.MachineName := '\\PC15';
Share1.ShareName       := 'Transfer'
Share1.ShareComment := 'For file transfer';
```

TShare.ShareName

TShare

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.

Example

```
Share1.MachineName := '\\PC25';
Share1.ShareName := 'COMMON';
try
Edit1.Text := Share1.ShareComment;
except
Edit1.Text := 'Resource \\PC25\COMMON does not exist';
end;
```

TShare.ShareType

TShare

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.

TShare.Usages

TShare

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.

Example

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

TShareType type

TShare

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

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 TShare TUsageList

TUsage represents an item in a TUsageList.

Description

TUsageList holds a group of TUsage objects. TUsage objects are created by TShare 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 TShare 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 TShare

Clear

Delete

TUsageList properties

TUsageList

Count

Items

TUsageList type

Properties Methods TShare

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 TShare.Usages property instead to obtain a pointer to the list of sessions maintained by TShare component.

Example

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

TUsageList.Clear

TUsageList TShare

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 TShare

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 TShare

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 TShare

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

TUserMan 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

TUserMan

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

TUserMan component

Hierarchy Properties Methods

Usage

Windows NT only

TUserMan 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 TUserMan component.

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

TUserMan methods

TUserMan

GetServers

TUserMan.GlobalGroupComment

TUserMan

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.

Example

```
   UserMan1.MachineName := '\\MOON';
..UserMan1.GlobalGroupName := 'IMAP4 project';
..UserMan1.GlobalGroupComment := 'Members of IMAP4 project'
```

TUserMan.GlobalGroupMembers

TUserMan

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.

Example 1 - replacement of the list of members

```
var
  List : TStringList;
begin
  List := TStringList.Create;
  try
  List.Add('Sergei A. Hramchenko');
  List.Add('John Smith');
  Userman1.GlobalGroupMembers := List;
  finally
  List.Free;
  end;
end;
```

Example 2 - how to add a new member to the global group

```
  Userman1.GlobalGroupMembers.Add('Sergei A. Hramchenko');
```

Example 3 - how to delete a member from the global group

```
with Userman1.GlobalGroupMembers do  Delete(IndexOf('Sergei A. Hramchenko'));
```

TUserMan.GlobalGroupName

TUserMan

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.

Example

```
  UserMan1.MachineName := '\\MOON';
..UserMan1.GlobalGroupName := 'IMAP4 project';
```

TUserMan.GlobalGroups

TUserMan

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.

Example 1 - replacement of the list of global groups

```
var
  List : TStringList;
begin
  List := TStringList.Create;
  try
  List.Add('Administration');
  List.Add('Internet');
  List.Add('Operators');
  Userman1.GlobalGroups := List;
  finally
  List.Free;
  end;
end;
```

Example 2 - how to add a new global group

```
  Userman1.GlobalGroups.Add('Administration');
```

Example 3 - how to delete a global group

```
with Userman1.GlobalGroups do  Delete(IndexOf('Administration'));
```

TUserMan.MachineName

TUserMan

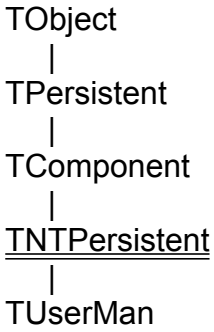
Property MachineName: string;

Description

MachineName contains the name of the target computer you are going to retrieve information from. If empty string is specified, you will get information from the local computer.

Example

```
UserMan1.MachineName := '\\PC21';
UserMan1.UserName := 'Administrator';
```


TObject
TPersistent
TComponent
TNTPersistent
TUserMan

TUserPriv type

TUserMan 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

TUserMan properties

TUserMan

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

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 lpLoadOrderGroup 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.

TUserMan.UserInfo

TUserMan

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.

Example

```
  UserMan1.MachineName := '';
  UserMan1.UserName := 'Administrator';
  Edit1.Text := UserMan1.UserInfo.Comment;
```

TUserMan.UserName

TUserMan

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.

Example

```
UserMan1.MachineName := '';
UserMan1.UserName := 'Administrator';
Edit1.Text := UserMan1.UserInfo.Comment;
```

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.

TUserMan.Users

TUserMan

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.

Example 1 - how to add a user

```
Userman1.Users.Add('Internet');
```

Example 2 - how to delete a user

```
with Userman1.Users do   Delete(IndexOf('Internet'));
```

Remark

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

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.

Example

```
UserMan1.UserInfo.Workstations := 'PC21,PC15';
```