

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;
```

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;
```

Contact information

TNTService component is the first in a row of components designed especially for Windows NT™

This is the share ware version. Full version with source code is available at \$30.

E-mail contact@risq.belcaf.minsk.by

Visit us at <http://risq.belcaf.minsk.by>

Contact information



TNTService 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;

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;
```

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.

GetDependentServiceList method

TNTService

function GetDependentServiceList(AState: TServiceStates): TEnumList

Description

The GetDependentServicesList 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;
```

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

NTService 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;
```

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';
```

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;
```

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;

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.

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';
```


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 flsLocked > 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;
```

ControlService

GetServiceDisplayName

GetServiceKeyName

GetServiceList
GetDependentServiceList
TEnumList type

QueryServiceLockStatus

QueryServiceStatus

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;
```

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 SrtartType: TStartType

Example:

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


TControlAcceptedSet type

TNTService See also

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.

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;

Decsription

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.

TEventLog component

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.

TNTService

[Hierarchy](#)

[Properties](#)

[Methods](#)

Usage

WindowsNT 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 to control 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 shows you the main features of TNTService component.

TObject
|
TPersistent
|
TComponent
|
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

TQueryServiceLockStatus type

[TNTService](#) [See also](#)

type

```
TQueryServiceLockStatus = record  
  flsLocked: DWORD;  
  lpLockOwner: PAnsiChar;  
  dwLockDuration: DWORD;  
end;
```

Description

flsLocked

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.

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;
  DisplayName:      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

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.

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 IpLoadOrderGroup 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.

