NSSM - the Non-Sucking Service Manager

Managing services from the command line

nssm's core functionality has always been available from the command line.

Service installation

```
nssm install <servicename>
nssm install <servicename> <program>
nssm install <servicename> <program> [<arguments>]
```

By default the service's startup directory will be set to the directory containing the `program`. The startup directory can be overridden after the service has been installed.

```
nssm set <servicename> AppDirectory <path>
```

Service removal

```
nssm remove
nssm remove <servicename>
nssm remove <servicename> confirm
```

Service management

As of version 2.22, *nssm* offers basic service management functionality. *nssm* will also accept a service `displayname` anywhere that a `servicename` is expected, since Windows does not allow a service to have a name or display name which conflicts with either the name or display name of another service. Both the service name (also called the service key name) and its display name uniquely identify a service.

Starting and stopping a service

```
nssm start <servicename>
nssm stop <servicename>
nssm restart <servicename>
```

Querying a service's status

```
nssm status <servicename>
```

Sending controls to services

```
nssm pause <servicename>
```

```
nssm continue <servicename>
nssm rotate <servicename>
```

`nssm rotate` triggers on-demand rotation for *nssm* services with I/O redirection and online rotation enabled. *nssm* accepts user-defined control 128 as a cue to begin output file rotation. Non-*nssm* services might respond to control 128 in their own way (or ignore it, or crash).

Service editing

As of version 2.22, *all* parameters understood by *nssm* can be queried or configured on the command line. A subset of system parameters can also be queried and, in some cases, modified.

General syntax

Parameters can usually be queried as follows.

```
nssm get <servicename> <parameter>
```

Some parameters are ambiguous and require a subparameter. See below.

```
nssm get <servicename> <parameter> <subparameter>
```

Parameters can usually be set in a similar way.

```
nssm set <servicename> <parameter> <value>
nssm set <servicename> <parameter> <subparameter> <value>
```

Most parameters can be reset to their defaults, which is equivalent to removing the associated registry entry.

```
nssm reset <servicename> <parameter>
nssm reset <servicename> <parameter> <subparameter>
```

As a convenience, *nssm* will accept additional arguments beyond the `value` required, and concatenate them together, separated by single spaces. Thus the following two invocations are identical:

```
nssm set <servicename> AppParameters "-classpath C:\Classes"
nssm set <servicename> AppParameters -classpath C:\Classes
```

Parameters

A `parameter` is usually a string with the same name as the registry entry which controls the associated functionality. So, for example, the following command sets the startup directory for a service:

```
nssm set <servicename> AppDirectory <path>
```

Values

Most parameters are configured by setting the same value as is documented for the

associated registry entry. To enable file rotation, for example, you would use the following command:

```
nssm set <servicename> AppRotation 1
```

See below for a list of parameters whose values are set differently.

Native parameters

Certain parameters configure properties of the service itself rather than the behaviour of *nssm*. They too are named after their associated registry values.

- **DependOnGroup:** Load order groups whose members must start before the service can start.
- **DependOnService:** Services which must start before the service can start.
- **Description:** The service's description
- **DisplayName:** The service's display name, eg *Application Layer Gateway Service*. This is the name shown under the *Name* column in *services.msc*.
- **ImagePath:** Path to the service executable, eg *C:\Windows\System32\alg.exe*. For *nssm* services, this will be the path to *nssm.exe*.
- **ObjectName:** The name of the user account under which the service runs. The default is *LOCALSYSTEM*.
- **Name:** The service key name, eg *ALG*. The key name cannot be changed. You can use `nssm get <displayname> Name` to find out the key name of a service.
- **Start:** The service's startup type, eg *Automatic*.
- **Type:** The service type. *nssm* can only edit services of type `SERVICE_WIN32_OWN_PROCESS`.

Non-standard parameters

- When used with `nssm get`, **AppEnvironment** and **AppEnvironmentExtra** accept an optional subparameter. If no subparameter is given, `nssm get` will print all configured environment variables, one per line in the form *KEY=VALUE*. If a subparameter is given, `nssm get` will print the value configured for the named environment variable, or the empty string if that variable is not present in the environment block.

  For example, suppose that **AppEnvironmentExtra** were configured with two variables, *CLASSPATH=C:\Classes* and *TEMP=C:\Temp*. The following invocation:

  ```
  nssm get <servicename> AppEnvironmentExtra
  ```

  would print:

  ```
  CLASSPATH=C:\Classes
  TEMP=C:\Temp
  ```

Whereas the syntax below:

```
nssm get <servicename> AppEnvironmentExtra CLASSPATH
```

would print:

```
C:\Classes
```

When setting an environment block with `nssm set`, each variable should be specified as a *KEY=VALUE* pair in a separate argument. For example:

```
nssm set <servicename> AppEnvironmentExtra CLASSPATH=C:\Classes TEMP=C:\Temp
```

- The **AppExit** parameter requires a subparameter specifying the exit code to get or set. The default action can be specified with the string *Default*.

  For example, to get the default exit action for a service you should run:

  ```
  nssm get <servicename> AppExit Default
  ```

  To get the exit action when the application exits with exit code 2, run:

  ```
  nssm get <servicename> AppExit 2
  ```

  Note that if no explicit action is configured for a specified exit code, *nssm* will print the default exit action.

  To set configure the service to stop when the application exits with an exit code of 2, run:

  ```
  nssm set <servicename> AppExit 2 Exit
  ```

- The **AppPriority** parameter takes a priority class constant as specified in the `SetPriorityClass()` documentation. Valid priorities are:

  - *REALTIME_PRIORITY_CLASS*
  - *HIGH_PRIORITY_CLASS*
  - *ABOVE_NORMAL_PRIORITY_CLASS*
  - *NORMAL_PRIORITY_CLASS*
  - *BELOW_NORMAL_PRIORITY_CLASS*
  - *IDLE_PRIORITY_CLASS*

Non-standard native parameters

- When used with `nssm set`, the **DependOnGroup** and **DependOnService** parameters treat each subsequent command line argument as a dependency group or service.

  Groups can be specified with or without the `SC_GROUP_IDENTIFIER` prefix (the + symbol). Services can be specified via their key name or display name.

  The following two invocations are equivalent:

  ```
  nssm set <servicename> DependOnService RpcSS LanmanWorkstation
  ```

  ```
  nssm set <servicename> DependOnService "Remote Procedure Call (RPC)" LanmanWorkstation
  ```

Groups will always be prefixed by the `SC_GROUP_IDENTIFIER` when queried with `nssm get`.

- When used with `nssm set`, the **ObjectName** parameter requires an additional argument specifying the password of the user which will run the service.

  To retrieve the username, run:

  ```
  nssm get <servicename> ObjectName
  ```

  To set the username and password, run:

  ```
  nssm set <servicename> ObjectName <username> <password>
  ```

  Note that the rules of argument concatenation still apply. The following invocation will have the expected effect:

  ```
  nssm set <servicename> ObjectName <username> correct horse battery staple
  ```

  If you absolutely must configure an account with a blank password, run:

  ```
  nssm set <servicename> ObjectName <username> ""
  ```

- Valid values for the **Start** parameter are:

  - *SERVICE_AUTO_START:* Automatic startup at boot.
  - *SERVICE_DELAYED_START:* Delayed startup at boot.
  - *SERVICE_DEMAND_START:* Manual startup.
  - *SERVICE_DISABLED:* Service is disabled.

  Note that *SERVICE_DELAYED_START* is not supported on versions of Windows prior to Vista. *nssm* will set the service to automatic startup if delayed start is unavailable.

- The **Type** parameter is used to query or set the service type. *nssm* recognises all currently documented service types but will only allow setting one of two types:

  - *SERVICE_WIN32_OWN_PROCESS:* A standalone service. This is the default.
  - *SERVICE_INTERACTIVE_PROCESS:* A service which can interact with the desktop.

  A service may only be configured as interactive if it runs under the *LOCALSYSTEM* account. To guarantee success when attempting to configure an interactive service, run two commands in sequence:

  ```
  nssm reset <servicename> ObjectName
  nssm set <servicename> Type SERVICE_INTERACTIVE_PROCESS
  ```