**[MS-SCMR]:**

**Service Control Manager Remote Protocol**

Intellectual Property Rights Notice for Open Specifications Documentation

* **Technical Documentation.** Microsoft publishes Open Specifications documentation (“this documentation”) for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
* **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
* **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
* **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft [Open Specifications Promise](http://go.microsoft.com/fwlink/?LinkId=214445) or the [Microsoft Community Promise](http://go.microsoft.com/fwlink/?LinkId=214448). If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting [iplg@microsoft.com](mailto:iplg@microsoft.com).
* **License Programs**. To see all of the protocols in scope under a specific license program and the associated patents, visit the [Patent Map](https://msdn.microsoft.com/en-us/openspecifications/dn750984).
* **Trademarks**. The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit [www.microsoft.com/trademarks](http://www.microsoft.com/trademarks).
* **Fictitious Names**. The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

**Reservation of Rights**. All other rights are reserved, and this notice does not grant any rights other than as specifically described above, whether by implication, estoppel, or otherwise.

**Tools**. The Open Specifications documentation does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments, you are free to take advantage of them. Certain Open Specifications documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

**Support.** For questions and support, please contact [dochelp@microsoft.com](mailto:dochelp@microsoft.com).

**Revision Summary**

| Date | Revision History | Revision Class | Comments |
| --- | --- | --- | --- |
| 5/11/2007 | 1.0 | Major | Version 1.0 release |
| 6/1/2007 | 1.0.1 | Editorial | Changed language and formatting in the technical content. |
| 7/3/2007 | 1.0.2 | Editorial | Changed language and formatting in the technical content. |
| 8/10/2007 | 1.1 | Minor | Revised content based on feedback. |
| 9/28/2007 | 1.2 | Minor | Revised content based on feedback. |
| 10/23/2007 | 1.2.1 | Editorial | Changed language and formatting in the technical content. |
| 1/25/2008 | 1.2.2 | Editorial | Changed language and formatting in the technical content. |
| 3/14/2008 | 2.0 | Major | Updated and revised the technical content. |
| 6/20/2008 | 3.0 | Major | Updated and revised the technical content. |
| 7/25/2008 | 3.0.1 | Editorial | Changed language and formatting in the technical content. |
| 8/29/2008 | 3.1 | Minor | Clarified the meaning of the technical content. |
| 10/24/2008 | 3.1.1 | Editorial | Changed language and formatting in the technical content. |
| 12/5/2008 | 4.0 | Major | Updated and revised the technical content. |
| 1/16/2009 | 5.0 | Major | Updated and revised the technical content. |
| 2/27/2009 | 6.0 | Major | Updated and revised the technical content. |
| 4/10/2009 | 7.0 | Major | Updated and revised the technical content. |
| 5/22/2009 | 8.0 | Major | Updated and revised the technical content. |
| 7/2/2009 | 9.0 | Major | Updated and revised the technical content. |
| 8/14/2009 | 10.0 | Major | Updated and revised the technical content. |
| 9/25/2009 | 11.0 | Major | Updated and revised the technical content. |
| 11/6/2009 | 12.0 | Major | Updated and revised the technical content. |
| 12/18/2009 | 13.0 | Major | Updated and revised the technical content. |
| 1/29/2010 | 13.1 | Minor | Clarified the meaning of the technical content. |
| 3/12/2010 | 14.0 | Major | Updated and revised the technical content. |
| 4/23/2010 | 15.0 | Major | Updated and revised the technical content. |
| 6/4/2010 | 16.0 | Major | Updated and revised the technical content. |
| 7/16/2010 | 17.0 | Major | Updated and revised the technical content. |
| 8/27/2010 | 18.0 | Major | Updated and revised the technical content. |
| 10/8/2010 | 18.1 | Minor | Clarified the meaning of the technical content. |
| 11/19/2010 | 19.0 | Major | Updated and revised the technical content. |
| 1/7/2011 | 19.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 2/11/2011 | 20.0 | Major | Updated and revised the technical content. |
| 3/25/2011 | 21.0 | Major | Updated and revised the technical content. |
| 5/6/2011 | 22.0 | Major | Updated and revised the technical content. |
| 6/17/2011 | 22.1 | Minor | Clarified the meaning of the technical content. |
| 9/23/2011 | 22.1 | None | No changes to the meaning, language, or formatting of the technical content. |
| 12/16/2011 | 23.0 | Major | Updated and revised the technical content. |
| 3/30/2012 | 23.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 7/12/2012 | 24.0 | Major | Updated and revised the technical content. |
| 10/25/2012 | 24.1 | Minor | Clarified the meaning of the technical content. |
| 1/31/2013 | 25.0 | Major | Updated and revised the technical content. |
| 8/8/2013 | 26.0 | Major | Updated and revised the technical content. |
| 11/14/2013 | 26.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 2/13/2014 | 26.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 5/15/2014 | 26.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 6/30/2015 | 27.0 | Major | Significantly changed the technical content. |
| 10/16/2015 | 27.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 7/14/2016 | 27.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 6/1/2017 | 27.0 | None | No changes to the meaning, language, or formatting of the technical content. |

Table of Contents

[1 Introduction 7](#_Toc483457048)

[1.1 Glossary 7](#_Toc483457049)

[1.2 References 10](#_Toc483457050)

[1.2.1 Normative References 10](#_Toc483457051)

[1.2.2 Informative References 10](#_Toc483457052)

[1.3 Overview 11](#_Toc483457053)

[1.4 Relationship to Other Protocols 11](#_Toc483457054)

[1.5 Prerequisites/Preconditions 11](#_Toc483457055)

[1.6 Applicability Statement 11](#_Toc483457056)

[1.7 Versioning and Capability Negotiation 11](#_Toc483457057)

[1.8 Vendor-Extensible Fields 12](#_Toc483457058)

[1.9 Standards Assignments 12](#_Toc483457059)

[2 Messages 13](#_Toc483457060)

[2.1 Transport 13](#_Toc483457061)

[2.1.1 Server 13](#_Toc483457062)

[2.1.2 Client 13](#_Toc483457063)

[2.2 Common Data Types 13](#_Toc483457064)

[2.2.1 SECURITY\_INFORMATION 14](#_Toc483457065)

[2.2.2 SVCCTL\_HANDLEA 14](#_Toc483457066)

[2.2.3 SVCCTL\_HANDLEW 14](#_Toc483457067)

[2.2.4 SC\_RPC\_HANDLE 14](#_Toc483457068)

[2.2.5 SC\_RPC\_LOCK 15](#_Toc483457069)

[2.2.6 SC\_NOTIFY\_RPC\_HANDLE 15](#_Toc483457070)

[2.2.7 BOUNDED\_DWORD\_4K 15](#_Toc483457071)

[2.2.8 BOUNDED\_DWORD\_8K 15](#_Toc483457072)

[2.2.9 BOUNDED\_DWORD\_256K 15](#_Toc483457073)

[2.2.10 ENUM\_SERVICE\_STATUSA 16](#_Toc483457074)

[2.2.11 ENUM\_SERVICE\_STATUSW 16](#_Toc483457075)

[2.2.12 ENUM\_SERVICE\_STATUS\_PROCESSA 16](#_Toc483457076)

[2.2.13 ENUM\_SERVICE\_STATUS\_PROCESSW 17](#_Toc483457077)

[2.2.14 QUERY\_SERVICE\_CONFIGA 17](#_Toc483457078)

[2.2.15 QUERY\_SERVICE\_CONFIGW 19](#_Toc483457079)

[2.2.16 QUERY\_SERVICE\_LOCK\_STATUSA 21](#_Toc483457080)

[2.2.17 QUERY\_SERVICE\_LOCK\_STATUSW 21](#_Toc483457081)

[2.2.18 SC\_ACTION\_TYPE 21](#_Toc483457082)

[2.2.19 SC\_ACTION 22](#_Toc483457083)

[2.2.20 SC\_ENUM\_TYPE 22](#_Toc483457084)

[2.2.21 SC\_RPC\_CONFIG\_INFOA 22](#_Toc483457085)

[2.2.22 SC\_RPC\_CONFIG\_INFOW 23](#_Toc483457086)

[2.2.23 SC\_RPC\_NOTIFY\_PARAMS 24](#_Toc483457087)

[2.2.24 SC\_RPC\_NOTIFY\_PARAMS\_LIST 24](#_Toc483457088)

[2.2.25 SC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSA 25](#_Toc483457089)

[2.2.26 SC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSW 25](#_Toc483457090)

[2.2.27 SC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSA 25](#_Toc483457091)

[2.2.28 SC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSW 26](#_Toc483457092)

[2.2.29 SC\_STATUS\_TYPE 26](#_Toc483457093)

[2.2.30 SERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSA 26](#_Toc483457094)

[2.2.31 SERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSW 29](#_Toc483457095)

[2.2.32 SERVICE\_CONTROL\_STATUS\_REASON\_OUT\_PARAMS 31](#_Toc483457096)

[2.2.33 SERVICE\_DELAYED\_AUTO\_START\_INFO 31](#_Toc483457097)

[2.2.34 SERVICE\_DESCRIPTIONA 32](#_Toc483457098)

[2.2.35 SERVICE\_DESCRIPTIONW 32](#_Toc483457099)

[2.2.36 SERVICE\_DESCRIPTION\_WOW64 32](#_Toc483457100)

[2.2.37 SERVICE\_FAILURE\_ACTIONS\_WOW64 32](#_Toc483457101)

[2.2.38 SERVICE\_REQUIRED\_PRIVILEGES\_INFO\_WOW64 33](#_Toc483457102)

[2.2.39 SERVICE\_FAILURE\_ACTIONSA 33](#_Toc483457103)

[2.2.40 SERVICE\_FAILURE\_ACTIONSW 34](#_Toc483457104)

[2.2.41 SERVICE\_FAILURE\_ACTIONS\_FLAG 34](#_Toc483457105)

[2.2.42 SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS 34](#_Toc483457106)

[2.2.43 SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_1 35](#_Toc483457107)

[2.2.44 SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_2 36](#_Toc483457108)

[2.2.45 SERVICE\_PRESHUTDOWN\_INFO 37](#_Toc483457109)

[2.2.46 SERVICE\_SID\_INFO 38](#_Toc483457110)

[2.2.47 SERVICE\_STATUS 38](#_Toc483457111)

[2.2.48 SERVICE\_RPC\_REQUIRED\_PRIVILEGES\_INFO 40](#_Toc483457112)

[2.2.49 SERVICE\_STATUS\_PROCESS 41](#_Toc483457113)

[2.2.50 STRING\_PTRSA 43](#_Toc483457114)

[2.2.51 STRING\_PTRSW 43](#_Toc483457115)

[2.2.52 SERVICE\_TRIGGER\_SPECIFIC\_DATA\_ITEM 43](#_Toc483457116)

[2.2.53 SERVICE\_TRIGGER 44](#_Toc483457117)

[2.2.54 SERVICE\_TRIGGER\_INFO 46](#_Toc483457118)

[2.2.55 SERVICE\_PREFERRED\_NODE\_INFO 47](#_Toc483457119)

[2.2.56 svcctl Interface Constants 47](#_Toc483457120)

[2.2.57 Common Error Codes 48](#_Toc483457121)

[3 Protocol Details 49](#_Toc483457122)

[3.1 Server Details 49](#_Toc483457123)

[3.1.1 Abstract Data Model 49](#_Toc483457124)

[3.1.2 Timers 57](#_Toc483457125)

[3.1.3 Initialization 58](#_Toc483457126)

[3.1.4 Message Processing Events and Sequencing Rules 58](#_Toc483457127)

[3.1.4.1 RCloseServiceHandle (Opnum 0) 63](#_Toc483457128)

[3.1.4.2 RControlService (Opnum 1) 63](#_Toc483457129)

[3.1.4.3 RDeleteService (Opnum 2) 66](#_Toc483457130)

[3.1.4.4 RLockServiceDatabase (Opnum 3) 66](#_Toc483457131)

[3.1.4.5 RQueryServiceObjectSecurity (Opnum 4) 67](#_Toc483457132)

[3.1.4.6 RSetServiceObjectSecurity (Opnum 5) 68](#_Toc483457133)

[3.1.4.7 RQueryServiceStatus (Opnum 6) 69](#_Toc483457134)

[3.1.4.8 RSetServiceStatus (Opnum 7) 70](#_Toc483457135)

[3.1.4.9 RUnlockServiceDatabase (Opnum 8) 72](#_Toc483457136)

[3.1.4.10 RNotifyBootConfigStatus (Opnum 9) 72](#_Toc483457137)

[3.1.4.11 RChangeServiceConfigW (Opnum 11) 73](#_Toc483457138)

[3.1.4.12 RCreateServiceW (Opnum 12) 77](#_Toc483457139)

[3.1.4.13 REnumDependentServicesW (Opnum 13) 81](#_Toc483457140)

[3.1.4.14 REnumServicesStatusW (Opnum 14) 82](#_Toc483457141)

[3.1.4.15 ROpenSCManagerW (Opnum 15) 84](#_Toc483457142)

[3.1.4.16 ROpenServiceW (Opnum 16) 85](#_Toc483457143)

[3.1.4.17 RQueryServiceConfigW (Opnum 17) 86](#_Toc483457144)

[3.1.4.18 RQueryServiceLockStatusW (Opnum 18) 87](#_Toc483457145)

[3.1.4.19 RStartServiceW (Opnum 19) 88](#_Toc483457146)

[3.1.4.20 RGetServiceDisplayNameW (Opnum 20) 90](#_Toc483457147)

[3.1.4.21 RGetServiceKeyNameW (Opnum 21) 91](#_Toc483457148)

[3.1.4.22 RChangeServiceConfigA (Opnum 23) 92](#_Toc483457149)

[3.1.4.23 RCreateServiceA (Opnum 24) 95](#_Toc483457150)

[3.1.4.24 REnumDependentServicesA (Opnum 25) 99](#_Toc483457151)

[3.1.4.25 REnumServicesStatusA (Opnum 26) 101](#_Toc483457152)

[3.1.4.26 ROpenSCManagerA (Opnum 27) 103](#_Toc483457153)

[3.1.4.27 ROpenServiceA (Opnum 28) 104](#_Toc483457154)

[3.1.4.28 RQueryServiceConfigA (Opnum 29) 105](#_Toc483457155)

[3.1.4.29 RQueryServiceLockStatusA (Opnum 30) 106](#_Toc483457156)

[3.1.4.30 RStartServiceA (Opnum 31) 106](#_Toc483457157)

[3.1.4.31 RGetServiceDisplayNameA (Opnum 32) 108](#_Toc483457158)

[3.1.4.32 RGetServiceKeyNameA (Opnum 33) 109](#_Toc483457159)

[3.1.4.33 REnumServiceGroupW (Opnum 35) 110](#_Toc483457160)

[3.1.4.34 RChangeServiceConfig2A (Opnum 36) 112](#_Toc483457161)

[3.1.4.35 RChangeServiceConfig2W (Opnum 37) 113](#_Toc483457162)

[3.1.4.36 RQueryServiceConfig2A (Opnum 38) 114](#_Toc483457163)

[3.1.4.37 RQueryServiceConfig2W (Opnum 39) 116](#_Toc483457164)

[3.1.4.38 RQueryServiceStatusEx (Opnum 40) 118](#_Toc483457165)

[3.1.4.39 REnumServicesStatusExA (Opnum 41) 119](#_Toc483457166)

[3.1.4.40 REnumServicesStatusExW (Opnum 42) 121](#_Toc483457167)

[3.1.4.41 RCreateServiceWOW64A (Opnum 44) 124](#_Toc483457168)

[3.1.4.42 RCreateServiceWOW64W (Opnum 45) 128](#_Toc483457169)

[3.1.4.43 RNotifyServiceStatusChange (Opnum 47) 131](#_Toc483457170)

[3.1.4.44 RGetNotifyResults (Opnum 48) 133](#_Toc483457171)

[3.1.4.45 RCloseNotifyHandle (Opnum 49) 133](#_Toc483457172)

[3.1.4.46 RControlServiceExA (Opnum 50) 134](#_Toc483457173)

[3.1.4.47 RControlServiceExW (Opnum 51) 137](#_Toc483457174)

[3.1.4.48 RQueryServiceConfigEx (Opnum 56) 139](#_Toc483457175)

[3.1.5 Timer Events 140](#_Toc483457176)

[3.1.6 Other Local Events 140](#_Toc483457177)

[3.1.7 Conversion Between ANSI and Unicode String Formats 140](#_Toc483457178)

[3.2 RPC Runtime Check Notes 141](#_Toc483457179)

[4 Protocol Examples 142](#_Toc483457180)

[5 Security 143](#_Toc483457181)

[5.1 Security Considerations for Implementers 143](#_Toc483457182)

[5.2 Index of Security Parameters 143](#_Toc483457183)

[6 Appendix A: Full IDL 144](#_Toc483457184)

[7 Appendix B: Product Behavior 158](#_Toc483457185)

[8 Change Tracking 165](#_Toc483457186)

[9 Index 166](#_Toc483457187)

# Introduction

The Service Control Manager Remote Protocol is a [**remote procedure call (RPC)**](#gt_8a7f6700-8311-45bc-af10-82e10accd331)–based client/server protocol that is used for remotely managing the [**Service Control Manager (SCM)**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84). The SCM is an RPC server that enables service configuration and control of [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) programs. For more information, see [[MSDN-WINSVC]](https://go.microsoft.com/fwlink/?LinkId=90701).

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

## Glossary

This document uses the following terms:

**access control entry (ACE)**: An entry in an access control list (ACL) that contains a set of user rights and a [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) that identifies a principal for whom the rights are allowed, denied, or audited.

**American National Standards Institute (ANSI) character set**: A character set defined by a [**code page**](#gt_210637d9-9634-4652-a935-ded3cd434f38) approved by the American National Standards Institute (ANSI). The term "ANSI" as used to signify Windows code pages is a historical reference and a misnomer that persists in the Windows community. The source of this misnomer stems from the fact that the Windows code page 1252 was originally based on an ANSI draft, which became International Organization for Standardization (ISO) Standard 8859-1 [[ISO/IEC-8859-1]](https://go.microsoft.com/fwlink/?LinkId=90689). In Windows, the ANSI character set can be any of the following code pages: 1252, 1250, 1251, 1253, 1254, 1255, 1256, 1257, 1258, 874, 932, 936, 949, or 950. For example, "ANSI application" is usually a reference to a non-[**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) or code-page-based application. Therefore, "ANSI character set" is often misused to refer to one of the character sets defined by a Windows code page that can be used as an active system code page; for example, character sets defined by code page 1252 or character sets defined by code page 950. Windows is now based on [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8), so the use of ANSI character sets is strongly discouraged unless they are used to interoperate with legacy applications or legacy data.

**authentication level**: A numeric value indicating the level of authentication or message protection that [**remote procedure call (RPC)**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) will apply to a specific message exchange. For more information, see [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 13.1.2.1 and [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15).

**Authentication Service (AS)**: A service that issues ticket granting tickets (TGTs), which are used for authenticating principals within the realm or domain served by the [**Authentication Service**](#gt_ffda2ae0-9528-42a7-ac1a-9d42d40674f7).

**code page**: An ordered set of characters of a specific script in which a numerical index (code-point value) is associated with each character. Code pages are a means of providing support for character sets and keyboard layouts used in different countries. Devices such as the display and keyboard can be configured to use a specific code page and to switch from one code page (such as the United States) to another (such as Portugal) at the user's request.

**delayed start group**: A service group initialized following a delay after the initial system boot for the purpose of improving system-boot performance.

**device interface class**: A way of exporting device and driver functionality to other components, including other drivers and user-mode applications. A driver can register a [**device interface class**](#gt_0b3a092c-d22a-46f6-9111-dfa1506a8661), and then enable an instance of the class for each device object to which user-mode I/O requests might be sent. On the highest level, a [**device interface class**](#gt_0b3a092c-d22a-46f6-9111-dfa1506a8661) is a grouping of devices by functionality. Each [**device interface class**](#gt_0b3a092c-d22a-46f6-9111-dfa1506a8661) is associated with a [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1). Vendors can create and define their own [**GUIDs**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) for [**device interface classes**](#gt_0b3a092c-d22a-46f6-9111-dfa1506a8661).

**discretionary access control list (DACL)**: An access control list (ACL) that is controlled by the owner of an object and that specifies the access particular users or groups can have to the object.

**globally unique identifier (GUID)**: A term used interchangeably with [**universally unique identifier (UUID)**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) in Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the value. Specifically, the use of this term does not imply or require that the algorithms described in [[RFC4122]](https://go.microsoft.com/fwlink/?LinkId=90460) or [C706] must be used for generating the [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1). See also [**universally unique identifier (UUID)**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3).

**Interface Definition Language (IDL)**: The International Standards Organization (ISO) standard language for specifying the interface for remote procedure calls. For more information, see [C706] section 4.

**load-order group**: A service group for the purpose of service loading and initialization ordering.

**Microsoft Interface Definition Language (MIDL)**: The Microsoft implementation and extension of the OSF-DCE [**Interface Definition Language (IDL)**](#gt_73177eec-4092-420f-92c5-60b2478df824). [**MIDL**](#gt_9c5903c1-1477-4181-b451-3ba1e34a0c0c) can also mean the [**Interface Definition Language (IDL)**](#gt_73177eec-4092-420f-92c5-60b2478df824) compiler provided by Microsoft. For more information, see [MS-RPCE].

**named pipe**: A named, one-way, or duplex pipe for communication between a pipe server and one or more pipe clients.

**NUMA Node**: An arrangement of processors and memory within a system supporting Non-Uniform Memory Access (NUMA) technology [[MSDN-NUMA]](https://go.microsoft.com/fwlink/?LinkId=151239).

**opnum**: An operation number or numeric identifier that is used to identify a specific [**remote procedure call (RPC)**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) method or a method in an interface. For more information, see [C706] section 12.5.2.12 or [MS-RPCE].

**remote procedure call (RPC)**: A context-dependent term commonly overloaded with three meanings. Note that much of the industry literature concerning RPC technologies uses this term interchangeably for any of the three meanings. Following are the three definitions: (\*) The runtime environment providing remote procedure call facilities. The preferred usage for this meaning is "RPC runtime". (\*) The pattern of request and response message exchange between two parties (typically, a client and a server). The preferred usage for this meaning is "RPC exchange". (\*) A single message from an exchange as defined in the previous definition. The preferred usage for this term is "RPC message". For more information about RPC, see [C706].

**RPC context handle**: A representation of state maintained between a remote procedure call (RPC) client and server. The state is maintained on the server on behalf of the client. An RPC context handle is created by the server and given to the client. The client passes the RPC context handle back to the server in method calls to assist in identifying the state. For more information, see [C706].

**RPC protocol sequence**: A character string that represents a valid combination of a [**remote procedure call (RPC)**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) protocol, a network layer protocol, and a transport layer protocol, as described in [C706] and [MS-RPCE].

**RPC server**: A computer on the network that waits for messages, processes them when they arrive, and sends responses using RPC as its transport acts as the responder during a remote procedure call (RPC) exchange.

**RPC transport**: The underlying network services used by the remote procedure call (RPC) runtime for communications between network nodes. For more information, see [C706] section 2.

**security descriptor**: A data structure containing the security information associated with a securable object. A [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) identifies an object's owner by its [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d). If access control is configured for the object, its [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) contains a [**discretionary access control list (DACL)**](#gt_d727f612-7a45-48e4-9d87-71735d62b321) with [**SIDs**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) for the security principals who are allowed or denied access. Applications use this structure to set and query an object's security status. The [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) is used to guard access to an object as well as to control which type of auditing takes place when the object is accessed. The [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350) format is specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.6; a string representation of [**security descriptors**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350), called SDDL, is specified in [MS-DTYP] section 2.5.1.

**security identifier (SID)**: An identifier for security principals that is used to identify an account or a group. Conceptually, the [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) is composed of an account authority portion (typically a domain) and a smaller integer representing an identity relative to the account authority, termed the relative identifier (RID). The [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) format is specified in [MS-DTYP] section 2.4.2; a string representation of [**SIDs**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) is specified in [MS-DTYP] section 2.4.2 and [[MS-AZOD]](%5bMS-AZOD%5d.pdf#Section_5a0a0a3ec7a742e1b5f2cc8d8bd9739e) section 1.1.1.2.

**Server Message Block (SMB)**: A protocol that is used to request file and print services from server systems over a network. The SMB protocol extends the CIFS protocol with additional security, file, and disk management support. For more information, see [[CIFS]](https://go.microsoft.com/fwlink/?LinkId=89836) and [[MS-SMB]](%5bMS-SMB%5d.pdf#Section_f210069c70864dc2885e861d837df688).

**service**: A program that is managed by the [**Service Control Manager (SCM)**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84). The execution of this program is governed by the rules defined by the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84).

**Service Control Manager (SCM)**: An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) server that enables configuration and control of [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) programs.

**service group**: A set of services that are grouped together for dependency or load-ordering purposes.

**service record**: An entry in the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database that contains the configuration information associated with a service.

**session key**: A relatively short-lived symmetric key (a cryptographic key negotiated by the client and the server based on a shared secret). A [**session key's**](#gt_4f67a585-fb00-4166-93e8-cf4abca8226d) lifespan is bounded by the session to which it is associated. A [**session key**](#gt_4f67a585-fb00-4166-93e8-cf4abca8226d) has to be strong enough to withstand cryptanalysis for the lifespan of the session.

**system access control list (SACL)**: An access control list (ACL) that controls the generation of audit messages for attempts to access a securable object. The ability to get or set an object's [**SACL**](#gt_c189801e-3752-4715-88f4-17804dad5782) is controlled by a privilege typically held only by system administrators.

**Unicode**: A character encoding standard developed by the Unicode Consortium that represents almost all of the written languages of the world. The [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) standard [[UNICODE5.0.0/2007]](https://go.microsoft.com/fwlink/?LinkId=154659) provides three forms (UTF-8, UTF-16, and UTF-32) and seven schemes (UTF-8, UTF-16, UTF-16 BE, UTF-16 LE, UTF-32, UTF-32 LE, and UTF-32 BE).

**universally unique identifier (UUID)**: A 128-bit value. UUIDs can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects in cross-process communication such as client and server interfaces, manager entry-point vectors, and [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) objects. UUIDs are highly likely to be unique. UUIDs are also known as [**globally unique identifiers (GUIDs)**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) and these terms are used interchangeably in the Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the UUID. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the UUID.

**well-known endpoint**: A preassigned, network-specific, stable address for a particular client/server instance. For more information, see [C706].

**MAY, SHOULD, MUST, SHOULD NOT, MUST NOT:** These terms (in all caps) are used as defined in [[RFC2119]](https://go.microsoft.com/fwlink/?LinkId=90317). All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

## References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the [Errata](http://msdn.microsoft.com/en-us/library/dn781092.aspx).

### Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact [dochelp@microsoft.com](mailto:dochelp@microsoft.com). We will assist you in finding the relevant information.

[C706] The Open Group, "DCE 1.1: Remote Procedure Call", C706, August 1997, [https://www2.opengroup.org/ogsys/catalog/c706](https://go.microsoft.com/fwlink/?LinkId=89824)

[MS-CIFS] Microsoft Corporation, "[Common Internet File System (CIFS) Protocol](%5bMS-CIFS%5d.pdf#Section_d416ff7cc536406ea9514f04b2fd1d2b)".

[MS-DTYP] Microsoft Corporation, "[Windows Data Types](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2)".

[MS-LSAD] Microsoft Corporation, "[Local Security Authority (Domain Policy) Remote Protocol](%5bMS-LSAD%5d.pdf#Section_1b5471ef4c334a91b079dfcbb82f05cc)".

[MS-RPCE] Microsoft Corporation, "[Remote Procedure Call Protocol Extensions](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)".

[MS-SMB] Microsoft Corporation, "[Server Message Block (SMB) Protocol](%5bMS-SMB%5d.pdf#Section_f210069c70864dc2885e861d837df688)".

[MS-UCODEREF] Microsoft Corporation, "[Windows Protocols Unicode Reference](%5bMS-UCODEREF%5d.pdf#Section_4a045e08fc294f22baf416f38c2825fb)".

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, [http://www.rfc-editor.org/rfc/rfc2119.txt](https://go.microsoft.com/fwlink/?LinkId=90317)

### Informative References

[MS-AZOD] Microsoft Corporation, "[Authorization Protocols Overview](%5bMS-AZOD%5d.pdf#Section_5a0a0a3ec7a742e1b5f2cc8d8bd9739e)".

[MSDN-CtrlSvcEx] Microsoft Corporation, "ControlServiceEx function", [http://msdn.microsoft.com/en-us/library/ms682110(VS.85).aspx](https://go.microsoft.com/fwlink/?LinkId=157502)

[MSDN-CtrlSvc] Microsoft Corporation, "ControlService function", [http://msdn.microsoft.com/en-us/library/ms682108(VS.85).asp](https://go.microsoft.com/fwlink/?LinkId=157503)

[MSDN-MIDL] Microsoft Corporation, "Microsoft Interface Definition Language (MIDL)", [http://msdn.microsoft.com/en-us/library/ms950375.aspx](https://go.microsoft.com/fwlink/?LinkId=90041)

[MSDN-NUMA] Microsoft Corporation, "NUMA Support", [http://msdn.microsoft.com/en-us/library/aa363804(VS.85).aspx](https://go.microsoft.com/fwlink/?LinkId=151239)

[MSDN-SetSvcStatus] Microsoft Corporation, "SetServiceStatus function", [http://msdn.microsoft.com/en-us/library/ms686241(VS.85).aspx](https://go.microsoft.com/fwlink/?LinkId=157504)

[MSDN-STARTSERVICE] Microsoft Corporation, "StartService function", [http://msdn.microsoft.com/en-us/library/ms686321.aspx](https://go.microsoft.com/fwlink/?LinkId=90137)

[MSDN-WinDriverKit] Microsoft Corporation, "Windows Driver Kit Introduction", [http://msdn.microsoft.com/en-us/library/ff556636(VS.85).aspx](https://go.microsoft.com/fwlink/?LinkId=151330)

[MSDN-WINSVC] Microsoft Corporation, "Services", [http://msdn.microsoft.com/en-us/library/ms685141.aspx](https://go.microsoft.com/fwlink/?LinkId=90701)

[SPNNAMES] Microsoft Corporation, "Name Formats for Unique SPNs", [http://msdn.microsoft.com/en-us/library/ms677601.aspx](https://go.microsoft.com/fwlink/?LinkId=90532)

## Overview

The Service Control Manager Remote Protocol is a client/server protocol used for configuring and controlling [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) programs running on a remote computer. A remote service management session begins with the client initiating the connection request to the server. If the server grants the request, the connection is established. The client can then make multiple requests to modify, query the configuration, or start and stop services on the server by using the same session until the session is terminated.

A typical Service Control Manager Remote Protocol session involves the client connecting to the server and requesting to open the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) on the server. If the server accepts the request, it responds with an [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) to the client. The client uses this RPC context handle to operate on the server. This usually involves sending another request to the server and specifying the type of operation to perform and any specific parameters associated with that operation. If the server accepts this request, it attempts to perform the specified operation and responds to the client with the result of the operation. After the client is finished operating on the server, it terminates the protocol by sending a request to close the RPC context handle.

The Service Control Manager Remote Protocol maintains an internal database to store service program configurations and state. The Service Control Manager Protocol has exclusive access to this internal database. On one operating system instance there is only one SCM and one corresponding SCM database. Any updates to this internal database are made only through the Service Control Manager Remote Protocol. SCM takes care of serializing all concurrent accesses to the SCM database. The SCM database is resident in memory; it is recreated every time the SCM restarts (after each reboot). Part of the SCM database is retrieved from persistent storage (all information regarding registered services) and partially nonpersistent (current active state of the services). The persistent information is modified by the SCM when a service is added, configured, or deleted. Any attempt to directly modify the persistent part of the database directly in the persistent storage is not a supported scenario and will result in possible inconsistencies. Finally, if SCM were to be forcefully terminated, the operating system will shut down and restart.

## Relationship to Other Protocols

The Service Control Manager Remote Protocol uses [**RPC as its transport**](#gt_c2eeb200-3cd0-4916-966e-d7d6bff1737a) protocol.

## Prerequisites/Preconditions

This protocol requires that the client and server be able to communicate via an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) connection, as specified in section [2.1](#Section_69bbbbfa6f2e449ea5bc47db341fe81b).

## Applicability Statement

This protocol is appropriate for managing a [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) management agent, such as an [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84), on a remote computer.

## Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

* **Supported Transports:** This protocol uses multiple [**RPC protocol sequences**](#gt_0c171cc7-e9c4-41b6-95a9-536db0042c7a), as specified in section [2.1](#Section_69bbbbfa6f2e449ea5bc47db341fe81b).
* **Security and Authentication Methods:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) server in this protocol uses either RPC\_C\_AUTHN\_GSS\_NEGOTIATE or RPC\_C\_AUTHN\_WINNT authorization. This is discussed in section 2.1.

## Vendor-Extensible Fields

None.

## Standards Assignments

The Service Control Manager Remote Protocol has no standards assignments, only private assignments made by Microsoft using allocation procedures specified in other protocols.

Microsoft has allocated to this protocol an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) interface [**universally unique identifier (UUID)**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) (using the procedure specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824)) and a [**named pipe**](#gt_34f1dfa8-b1df-4d77-aa6e-d777422f9dca) (as specified in [[MS-SMB]](%5bMS-SMB%5d.pdf#Section_f210069c70864dc2885e861d837df688)). The assignments are as follows.

| Parameter | Value |
| --- | --- |
| RPC interface UUID | {367ABB81-9844-35F1-AD32-98F038001003} |
| Named pipe | \PIPE\svcctl |

# Messages

The following sections specify how Service Control Manager Remote Protocol messages are transported and specify common data types.

## Transport

The Service Control Manager Remote Protocol MUST use [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) as the transport protocol.

### Server

The server interface is identified by [**UUID**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) 367ABB81-9844-35F1-AD32-98F038001003, version 2.0, using the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) [**well-known endpoint**](#gt_a7498f8e-e85d-473b-9fc9-d2fffdf71c8a) "\PIPE\svcctl". The server MUST use RPC over [**SMB**](#gt_09dbec39-5e75-4d9a-babf-1c9f1d499625), ncacn\_np or RPC over TCP, or ncacn\_ip\_tcp as the RPC protocol sequence to the RPC implementation, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). The server MUST specify the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) (0x9) or NT LAN Manager (NTLM) (0xA), or both, as the RPC [**Authentication Service**](#gt_ffda2ae0-9528-42a7-ac1a-9d42d40674f7) (as specified in [MS-RPCE]). See [MS-RPCE] section 3.3.1.5.2.2 and [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 13.

### Client

The client MUST use [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) over [**SMB**](#gt_09dbec39-5e75-4d9a-babf-1c9f1d499625), ncacn\_np (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)) or RPC over TCP, ncacn\_ip\_tcp (as specified in [MS-RPCE]) as the [**RPC protocol sequence**](#gt_0c171cc7-e9c4-41b6-95a9-536db0042c7a) to communicate with the server. The client MUST specify either "Simple and Protected GSS-API Negotiation Mechanism (SPNEGO)" (0x9) or "NT LAN Manager (NTLM)" (0xA), as specified in [MS-RPCE], as the [**Authentication Service**](#gt_ffda2ae0-9528-42a7-ac1a-9d42d40674f7). When using "SPNEGO" as the Authentication Service, the client SHOULD supply a service principal name (SPN) of "host/hostname" where hostname is the actual name of the server to which the client is connecting and host is the literal string "host/" (for more information, see [[SPNNAMES]](https://go.microsoft.com/fwlink/?LinkId=90532)).

The RPC client MAY use an [**authentication level**](#gt_bfb9708e-9d05-4f79-8969-ef63f73aa434) of RPC\_C\_AUTHN\_LEVEL\_PKT\_PRIVACY.[<1>](#Appendix_A_1" \o "Product behavior note 1)

## Common Data Types

In addition to [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) base types and definitions specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) and [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15), the following sections use these definitions, as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2). Unless specified, all characters are accepted for the strings described in each section.

* BOOL
* BYTE
* CHAR
* DWORD
* LPCSTR
* LPCWSTR
* LPWSTR
* PSTR
* UCHAR
* VOID
* WCHAR

The additional data types given in the following sections are defined in the [**MIDL**](#gt_9c5903c1-1477-4181-b451-3ba1e34a0c0c) specification of this RPC interface.

### SECURITY\_INFORMATION

The following bit flags indicate which components to include in a **SECURITY\_DESCRIPTOR** structure that clients and servers can use to specify access types.

| Value | Meaning |
| --- | --- |
| DACL\_SECURITY\_INFORMATION  0x00000004 | If set, the security descriptor MUST include the object's [**discretionary access control list (DACL)**](#gt_d727f612-7a45-48e4-9d87-71735d62b321). DACL information is specified in [[MS-AZOD]](%5bMS-AZOD%5d.pdf#Section_5a0a0a3ec7a742e1b5f2cc8d8bd9739e) section 1.1.1.3. |
| GROUP\_SECURITY\_INFORMATION  0x00000002 | If set, specifies the [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d), as defined in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.2, (LSAPR\_SID) of the object's primary group. Primary group information is specified in [MS-DTYP]. |
| OWNER\_SECURITY\_INFORMATION  0x00000001 | If set, specifies the security identifier (SID) (LSAPR\_SID) of the object's owner. |
| SACL\_SECURITY\_INFORMATION  0x00000008 | If set, the security descriptor MUST include the object's [**system access control list (SACL)**](#gt_c189801e-3752-4715-88f4-17804dad5782). SACL information is specified in [MS-AZOD] section 1.1.1.3. |

This type is declared as follows:

1. typedef unsigned long SECURITY\_INFORMATION;

### SVCCTL\_HANDLEA

An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle to the server, represented as an [**American National Standards Institute (ANSI) character set**](#gt_100cd8a6-5cb1-4895-9de6-e4a3c224a583) string. This ANSI string and all ANSI references in the rest of this document use the ANSI [**code page**](#gt_210637d9-9634-4652-a935-ded3cd434f38) specified by the operating system.

This type is declared as follows:

1. typedef [handle] LPSTR SVCCTL\_HANDLEA;

### SVCCTL\_HANDLEW

An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle represented as a [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string.

This type is declared as follows:

1. typedef [handle] wchar\_t\* SVCCTL\_HANDLEW;

### SC\_RPC\_HANDLE

Defines an [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) to the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) or a [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) on the server.

1. typedef [context\_handle] PVOID SC\_RPC\_HANDLE;
2. typedef SC\_RPC\_HANDLE\* LPSC\_RPC\_HANDLE;

### SC\_RPC\_LOCK

Defines an [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) to a locked [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database on the server.

1. typedef [context\_handle] PVOID SC\_RPC\_LOCK;
2. typedef SC\_RPC\_LOCK\* LPSC\_RPC\_LOCK;

### SC\_NOTIFY\_RPC\_HANDLE

Defines an [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) used to monitor changes on a [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) on the server.

1. typedef [context\_handle] PVOID SC\_NOTIFY\_RPC\_HANDLE;
2. typedef SC\_NOTIFY\_RPC\_HANDLE\* LPSC\_NOTIFY\_RPC\_HANDLE;

### BOUNDED\_DWORD\_4K

A 4-kilobyte ranged **DWORD** data type used for the size given by reference in an in/out parameter.

1. typedef [range(0, 1024 \* 4)] DWORD BOUNDED\_DWORD\_4K;
2. typedef BOUNDED\_DWORD\_4K\* LPBOUNDED\_DWORD\_4K;

BOUNDED\_DWORD\_4K

A 4-kilobyte ranged **DWORD** used for size given by reference in an in/out parameter.

LPBOUNDED\_DWORD\_4K

Pointer to a **BOUNDED\_DWORD\_4K**.

### BOUNDED\_DWORD\_8K

An 8-kilobyte ranged **DWORD** data type used for the size given by reference in an in/out parameter.

1. typedef [range(0, 1024 \* 8)] DWORD BOUNDED\_DWORD\_8K;
2. typedef BOUNDED\_DWORD\_8K\* LPBOUNDED\_DWORD\_8K;

BOUNDED\_DWORD\_8K

An 8-kilobyte ranged **DWORD** used for size given by reference in an in/out parameter.

LPBOUNDED\_DWORD\_8K

Pointer to a **BOUNDED\_DWORD\_8K**.

### BOUNDED\_DWORD\_256K

A 256-kilobyte ranged **DWORD** data type used for the size given by reference in an in/out parameter.

1. typedef [range(0, 1024 \* 256)]
2. DWORD BOUNDED\_DWORD\_256K;
3. typedef BOUNDED\_DWORD\_256K\* LPBOUNDED\_DWORD\_256K;

BOUNDED\_DWORD\_256K

A 256-kilobyte ranged **DWORD** used for size given by reference in an in/out parameter.

LPBOUNDED\_DWORD\_256K

Pointer to a **BOUNDED\_DWORD\_256K**.

### ENUM\_SERVICE\_STATUSA

The ENUM\_SERVICE\_STATUSA structure defines the name and status of a service in an [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database and returns information about the [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a). String values are stored in ANSI.

1. typedef struct \_ENUM\_SERVICE\_STATUSA {
2. LPSTR lpServiceName;
3. LPSTR lpDisplayName;
4. SERVICE\_STATUS ServiceStatus;
5. } ENUM\_SERVICE\_STATUSA,
6. \*LPENUM\_SERVICE\_STATUSA;

**lpServiceName:**  A pointer to a null-terminated string that names a service in an SCM database.

The forward slash, back slash, comma, and space characters are illegal in service names.

**lpDisplayName:**  A pointer to a null-terminated string that user interface programs use to identify the service.

**ServiceStatus:**  A [SERVICE\_STATUS (section 2.2.47)](#Section_4e91ff36ab5f49eda43da308e72b0b3c) structure that contains status information.

### ENUM\_SERVICE\_STATUSW

The ENUM\_SERVICE\_STATUSW structure defines the name and status of a service in an SCM database and returns information about the service. String values are stored in Unicode.

1. typedef struct \_ENUM\_SERVICE\_STATUSW {
2. LPWSTR lpServiceName;
3. LPWSTR lpDisplayName;
4. SERVICE\_STATUS ServiceStatus;
5. } ENUM\_SERVICE\_STATUSW,
6. \*LPENUM\_SERVICE\_STATUSW;

**lpServiceName:**  A pointer to a null-terminated string that names a service in an [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database.

The forward slash, back slash, comma, and space characters are illegal in service names.

**lpDisplayName:**  A pointer to a null-terminated string that user interface programs use to identify the [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a).

**ServiceStatus:**  A [SERVICE\_STATUS (section 2.2.47)](#Section_4e91ff36ab5f49eda43da308e72b0b3c) structure that contains status information.

### ENUM\_SERVICE\_STATUS\_PROCESSA

The ENUM\_SERVICE\_STATUS\_PROCESSA structure contains information used by the [REnumServicesStatusExA](#Section_f5512859cae94a109636eefeb0abd9a4) method to return the name of a [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) in an [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database. The structure also returns information about the service. String values are stored in ANSI.

1. typedef struct \_ENUM\_SERVICE\_STATUS\_PROCESSA {
2. LPSTR lpServiceName;
3. LPSTR lpDisplayName;
4. SERVICE\_STATUS\_PROCESS ServiceStatusProcess;
5. } ENUM\_SERVICE\_STATUS\_PROCESSA,
6. \*LPENUM\_SERVICE\_STATUS\_PROCESSA;

**lpServiceName:**  A pointer to a null-terminated string that names a service in an SCM database.

The forward slash, back slash, comma, and space characters are illegal in service names.

**lpDisplayName:**  A pointer to a null-terminated string that contains the display name of the service.

**ServiceStatusProcess:**  A [SERVICE\_STATUS\_PROCESS (section 2.2.49)](#Section_c2f0ab87eb1643718380ddf9cd29931e) structure that contains status information for the lpServiceName service.

### ENUM\_SERVICE\_STATUS\_PROCESSW

The ENUM\_SERVICE\_STATUS\_PROCESSW structure contains information used by the [REnumServicesStatusExW](#Section_fdb7df2e341e4dccad5fd16c2ac51466) method to return the name of a service in an SCM database. The structure also returns information about the service. String values are stored in Unicode.

1. typedef struct \_ENUM\_SERVICE\_STATUS\_PROCESSW {
2. LPWSTR lpServiceName;
3. LPWSTR lpDisplayName;
4. SERVICE\_STATUS\_PROCESS ServiceStatusProcess;
5. } ENUM\_SERVICE\_STATUS\_PROCESSW,
6. \*LPENUM\_SERVICE\_STATUS\_PROCESSW;

**lpServiceName:**  A pointer to a null-terminated string that names a service in an SCM database.

The forward slash, back slash, comma, and space characters are illegal in service names.

**lpDisplayName:**  A pointer to a null-terminated string that contains the display name of the service.

**ServiceStatusProcess:**  A [SERVICE\_STATUS\_PROCESS (section 2.2.49)](#Section_c2f0ab87eb1643718380ddf9cd29931e) structure that contains status information for the lpServiceName service.

### QUERY\_SERVICE\_CONFIGA

The QUERY\_SERVICE\_CONFIGA structure defines configuration information about an installed service. String values are stored in ANSI.

1. typedef struct \_QUERY\_SERVICE\_CONFIGA {
2. DWORD dwServiceType;
3. DWORD dwStartType;
4. DWORD dwErrorControl;
5. [string,range(0, 8 \* 1024)] LPSTR lpBinaryPathName;
6. [string,range(0, 8 \* 1024)] LPSTR lpLoadOrderGroup;
7. DWORD dwTagId;
8. [string,range(0, 8 \* 1024)] LPSTR lpDependencies;
9. [string,range(0, 8 \* 1024)] LPSTR lpServiceStartName;
10. [string,range(0, 8 \* 1024)] LPSTR lpDisplayName;
11. } QUERY\_SERVICE\_CONFIGA,
12. \*LPQUERY\_SERVICE\_CONFIGA;

**dwServiceType:**  The type of service. This member MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_KERNEL\_DRIVER  0x00000001 | A driver service. These are services that manage devices on the system. |
| SERVICE\_FILE\_SYSTEM\_DRIVER  0x00000002 | A file system driver service. These are services that manage file systems on the system. |
| SERVICE\_WIN32\_OWN\_PROCESS  0x00000010 | A service that runs in its own process. |
| SERVICE\_WIN32\_SHARE\_PROCESS  0x00000020 | A service that shares a process with other services. |

**dwStartType:**  Defines when to start the service. This member MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_BOOT\_START  0x00000000 | Starts the driver service when the system boots up. This value is valid only for driver services. |
| SERVICE\_SYSTEM\_START  0x00000001 | Starts the driver service when the system boots up. This value is valid only for driver services. The services marked SERVICE\_SYSTEM\_START are started after all SERVICE\_BOOT\_START services have been started. |
| SERVICE\_AUTO\_START  0x00000002 | A service started automatically by the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) during system startup. |
| SERVICE\_DEMAND\_START  0x00000003 | Starts the service when a client requests the SCM to start the service. |
| SERVICE\_DISABLED  0x00000004 | A service that cannot be started. Attempts to start the service result in the error code ERROR\_SERVICE\_DISABLED. |

**dwErrorControl:**  The severity of the error if this service fails to start during startup, and the action that the SCM takes if failure occurs.

| Value | Meaning |
| --- | --- |
| SERVICE\_ERROR\_IGNORE  0x00000000 | The SCM ignores the error and continues the startup operation. |
| SERVICE\_ERROR\_NORMAL  0x00000001 | The SCM logs the error in the event log and continues the startup operation. |
| SERVICE\_ERROR\_SEVERE  0x00000002 | The SCM logs the error in the event log. 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  0x00000003 | The SCM SHOULD log the error in the event log 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. |

**lpBinaryPathName:**  A pointer to a null-terminated string that contains the fully qualified path to the service binary file. The path MAY include arguments. If the path contains a space, it MUST be quoted so that it is correctly interpreted. For example, "d:\\my share\\myservice.exe" is specified as "\"d:\\my share\\myservice.exe\"".

**lpLoadOrderGroup:**  A pointer to a null-terminated string that names the [**service group**](#gt_8ee1e5c0-3886-409f-8707-197e6232042d) for load-ordering of which this service is a member. If the pointer is NULL or if it points to an empty string, the service does not belong to a group.

**dwTagId:**  A unique tag value for this service within the service group specified by the *lpLoadOrderGroup* parameter. A value of 0 indicates that the service has not been assigned a tag.

**lpDependencies:**  A pointer to an array of null-separated names of services or names of service groups that MUST start before this service. The array is doubly null-terminated. Service group names are prefixed with a "+" character (to distinguish them from service names). If the pointer is **NULL** or if it points to an empty string, the service has no dependencies. Cyclic dependency between services is not allowed. The character set is [**ANSI**](#gt_100cd8a6-5cb1-4895-9de6-e4a3c224a583). 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.

**lpServiceStartName:**  A pointer to a null-terminated string that contains the service name.

**lpDisplayName:**  A pointer to a null-terminated string that contains the service display name.

### QUERY\_SERVICE\_CONFIGW

The QUERY\_SERVICE\_CONFIGW structure defines configuration information about an installed service. String values are stored in Unicode.

1. typedef struct \_QUERY\_SERVICE\_CONFIGW {
2. DWORD dwServiceType;
3. DWORD dwStartType;
4. DWORD dwErrorControl;
5. [string,range(0, 8 \* 1024)] LPWSTR lpBinaryPathName;
6. [string,range(0, 8 \* 1024)] LPWSTR lpLoadOrderGroup;
7. DWORD dwTagId;
8. [string,range(0, 8 \* 1024)] LPWSTR lpDependencies;
9. [string,range(0, 8 \* 1024)] LPWSTR lpServiceStartName;
10. [string,range(0, 8 \* 1024)] LPWSTR lpDisplayName;
11. } QUERY\_SERVICE\_CONFIGW,
12. \*LPQUERY\_SERVICE\_CONFIGW;

**dwServiceType:**  The type of service. This member MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_KERNEL\_DRIVER  0x00000001 | A driver service. These are services that manage devices on the system. |
| SERVICE\_FILE\_SYSTEM\_DRIVER  0x00000002 | A file system driver service. These are services that manage file systems on the system. |
| SERVICE\_WIN32\_OWN\_PROCESS  0x00000010 | A service that runs in its own process. |
| SERVICE\_WIN32\_SHARE\_PROCESS  0x00000020 | A service that shares a process with other services. |

**dwStartType:**  Defines when to start the service. This member MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_BOOT\_START  0x00000000 | Starts the driver service when the system boots up. This value is valid only for driver services. |
| SERVICE\_SYSTEM\_START  0x00000001 | Starts the driver service when the system boots up. This value is valid only for driver services. The services marked SERVICE\_SYSTEM\_START are started after all SERVICE\_BOOT\_START services have been started. |
| SERVICE\_AUTO\_START  0x00000002 | A service started automatically by the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) during system startup. |
| SERVICE\_DEMAND\_START  0x00000003 | Starts the service when a client requests the SCM to start the service. |
| SERVICE\_DISABLED  0x00000004 | A service that cannot be started. Attempts to start the service result in the error code ERROR\_SERVICE\_DISABLED. |

**dwErrorControl:**  The severity of the error if this service fails to start during startup and the action the SCM takes if failure occurs.

| Value | Meaning |
| --- | --- |
| SERVICE\_ERROR\_IGNORE  0x00000000 | The SCM ignores the error and continues the startup operation. |
| SERVICE\_ERROR\_NORMAL  0x00000001 | The SCM logs the error in the event log and continues the startup operation. |
| SERVICE\_ERROR\_SEVERE  0x00000002 | The SCM logs the error in the event log. 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  0x00000003 | The SCM SHOULD log the error in the event log 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. |

**lpBinaryPathName:**  A pointer to a null-terminated string that contains the fully qualified path to the service binary file. The path MAY include arguments. If the path contains a space, it MUST be quoted so that it is correctly interpreted. For example, "d:\\my share\\myservice.exe" is specified as "\"d:\\my share\\myservice.exe\"".

**lpLoadOrderGroup:**  A pointer to a null-terminated string that names the [**service group**](#gt_8ee1e5c0-3886-409f-8707-197e6232042d) for load ordering of which this service is a member. If the pointer is **NULL** or if it points to an empty string, the service does not belong to a group.

**dwTagId:**  A unique tag value for this service in the service group. A value of 0 indicates that the service has not been assigned a tag.

**lpDependencies:**  A pointer to an array of null-separated names of services or service groups that MUST start before this service. The array is doubly null-terminated. Service group names are prefixed with a "+" character (to distinguish them from service names). If the pointer is NULL or if it points to an empty string, the service has no dependencies. Cyclic dependency between services is not allowed. The character set is Unicode. 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.

**lpServiceStartName:**  A pointer to a null-terminated string that contains the service start (key) name.

**lpDisplayName:**  A pointer to a null-terminated string that contains the service display name.

### QUERY\_SERVICE\_LOCK\_STATUSA

The QUERY\_SERVICE\_LOCK\_STATUSA structure defines information about the lock status of an [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database. String values are stored in ANSI.

1. typedef struct {
2. DWORD fIsLocked;
3. [string,range(0, 8 \* 1024)] char\* lpLockOwner;
4. DWORD dwLockDuration;
5. } QUERY\_SERVICE\_LOCK\_STATUSA,
6. \*LPQUERY\_SERVICE\_LOCK\_STATUSA;

**fIsLocked:**  The lock status of the database. If this member is nonzero, the database is locked. If it is 0, the database is unlocked.

**lpLockOwner:**  A pointer to a null-terminated string that contains the name of the user that acquired the lock.

**dwLockDuration:**  The elapsed time, in seconds, since the lock was first acquired.

### QUERY\_SERVICE\_LOCK\_STATUSW

The QUERY\_SERVICE\_LOCK\_STATUSW structure defines information about the lock status of an [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database. String values are stored in Unicode.

1. typedef struct \_QUERY\_SERVICE\_LOCK\_STATUSW {
2. DWORD fIsLocked;
3. [string,range(0, 8 \* 1024)] LPWSTR lpLockOwner;
4. DWORD dwLockDuration;
5. } QUERY\_SERVICE\_LOCK\_STATUSW,
6. \*LPQUERY\_SERVICE\_LOCK\_STATUSW;

**fIsLocked:**  The lock status of the database. If this member is nonzero, the database is locked. If it is 0, the database is unlocked.

**lpLockOwner:**  A pointer to a null-terminated string that contains the name of the user that acquired the lock.

**dwLockDuration:**  The elapsed time, in seconds, since the lock was first acquired.

### SC\_ACTION\_TYPE

The SC\_ACTION\_TYPE enumeration specifies action levels for the **Type** member of the [SC\_ACTION](#Section_5e3e9ca16c334c94bf36c5b75262d7d6) structure.

1. typedef [v1\_enum] enum \_SC\_ACTION\_TYPE
2. {
3. SC\_ACTION\_NONE = 0,
4. SC\_ACTION\_RESTART = 1,
5. SC\_ACTION\_REBOOT = 2,
6. SC\_ACTION\_RUN\_COMMAND = 3
7. } SC\_ACTION\_TYPE;

**SC\_ACTION\_NONE:** No action.

**SC\_ACTION\_RESTART:** Restart the service.

**SC\_ACTION\_REBOOT:** Reboot the computer.

**SC\_ACTION\_RUN\_COMMAND:** Run a command.

### SC\_ACTION

The SC\_ACTION structure defines an action that the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) can perform.

1. typedef struct {
2. SC\_ACTION\_TYPE Type;
3. DWORD Delay;
4. } SC\_ACTION,
5. \*LPSC\_ACTION;

**Type:**  The action to be performed. This member MUST be one of the values from the [SC\_ACTION\_TYPE (section 2.2.18)](#Section_7e3bf33ced3d4bc68781cc9fe092de9e) enumeration.

**Delay:**  The time, in milliseconds, to wait before performing the specified action.

### SC\_ENUM\_TYPE

The SC\_ENUM\_TYPE enumeration specifies information levels for the [REnumServicesStatusExA](#Section_f5512859cae94a109636eefeb0abd9a4) and [REnumServicesStatusExW](#Section_fdb7df2e341e4dccad5fd16c2ac51466) methods.

1. typedef [v1\_enum] enum
2. {
3. SC\_ENUM\_PROCESS\_INFO = 0
4. } SC\_ENUM\_TYPE;

**SC\_ENUM\_PROCESS\_INFO:** Information level.

### SC\_RPC\_CONFIG\_INFOA

The SC\_RPC\_CONFIG\_INFOA structure defines the service configuration based on a supplied level. String values are stored in ANSI.

1. typedef struct \_SC\_RPC\_CONFIG\_INFOA {
2. DWORD dwInfoLevel;
3. [switch\_is(dwInfoLevel)] union {
4. [case(1)]
5. LPSERVICE\_DESCRIPTIONA psd;
6. [case(2)]
7. LPSERVICE\_FAILURE\_ACTIONSA psfa;
8. [case(3)]
9. LPSERVICE\_DELAYED\_AUTO\_START\_INFO psda;
10. [case(4)]
11. LPSERVICE\_FAILURE\_ACTIONS\_FLAG psfaf;
12. [case(5)]
13. LPSERVICE\_SID\_INFO pssid;
14. [case(6)]
15. LPSERVICE\_RPC\_REQUIRED\_PRIVILEGES\_INFO psrp;
16. [case(7)]
17. LPSERVICE\_PRESHUTDOWN\_INFO psps;
18. [case(8)]
19. PSERVICE\_TRIGGER\_INFO psti;
20. [case(9)]
21. LPSERVICE\_PREFERRED\_NODE\_INFO pspn;
22. };
23. } SC\_RPC\_CONFIG\_INFOA;

**dwInfoLevel:**  A DWORD value that indicates the type of configuration information in the included data.

psd:  A structure that contains a description of the service, as specified in section [2.2.34](#Section_91a48e76b6b843eb9ea10757c4aa9260).

The following structures SHOULD[<2>](#Appendix_A_2" \o "Product behavior note 2) be available:

**psfa:**  A structure that contains a list of failure actions, as specified in section [2.2.39](#Section_a5b207c7114e48aea1c6ef79c1fe1854).

**psda:**  A structure that defines whether or not the service is part of the [**delayed start group**](#gt_09e2f255-29ce-44c5-82eb-a601ce39ec56), as specified in section [2.2.33](#Section_805b8296863d4d1e8ae8f639adf8c6cb).

**psfaf:**  A structure that defines if failure actions are queued when the service exits with a nonzero error code, as specified in section [2.2.41](#Section_9b244e2e82fc4c548f4fb19034faa2c4).

**pssid:**  A structure that defines the type of service SID, as specified in section [2.2.46](#Section_ea1a9acd4bb2473fae5b55969c5960fc).

**psrp:**  A structure that defines the privileges required by the service, as specified in section [2.2.48](#Section_2917121dbbe147d2ad557bce44a127e6).

**psps:**  A structure that defines the pre-shutdown settings for the service, as specified in section [2.2.45](#Section_6b55ad95df3e498fb107e77f831ba467).

**psti:**  A structure that defines the trigger settings for the service, as specified in section [2.2.54](#Section_a56f5538d13b49bfbd7c2c5b213fc0f1).

**pspn:**  A structure that defines the preferred node information for the service, as specified in section [2.2.55](#Section_3767259ff0544076acd39f7ecb79cab3).

### SC\_RPC\_CONFIG\_INFOW

The SC\_RPC\_CONFIG\_INFOW structure SHOULD[<3>](#Appendix_A_3" \o "Product behavior note 3) define, based on a supplied level, either the service configuration or a list of failure actions. String values are stored as Unicode.

1. typedef struct \_SC\_RPC\_CONFIG\_INFOW {
2. DWORD dwInfoLevel;
3. [switch\_is(dwInfoLevel)] union {
4. [case(1)]
5. LPSERVICE\_DESCRIPTIONW psd;
6. [case(2)]
7. LPSERVICE\_FAILURE\_ACTIONSW psfa;
8. [case(3)]
9. LPSERVICE\_DELAYED\_AUTO\_START\_INFO psda;
10. [case(4)]
11. LPSERVICE\_FAILURE\_ACTIONS\_FLAG psfaf;
12. [case(5)]
13. LPSERVICE\_SID\_INFO pssid;
14. [case(6)]
15. LPSERVICE\_RPC\_REQUIRED\_PRIVILEGES\_INFO psrp;
16. [case(7)]
17. LPSERVICE\_PRESHUTDOWN\_INFO psps;
18. [case(8)]
19. PSERVICE\_TRIGGER\_INFO psti;
20. [case(9)]
21. LPSERVICE\_PREFERRED\_NODE\_INFO pspn;
22. };
23. } SC\_RPC\_CONFIG\_INFOW;

**dwInfoLevel:**  A value that indicates the type of configuration information in the included data.

**psd:**  A structure that contains a description of the service, as specified in section [2.2.35](#Section_3de0acf0bd9b4b0ba37602036b16a83f).

**psfa:**  A structure that contains a list of failure actions, as specified in section [2.2.40](#Section_58032b711e5c4f2e854534b0f2e8c6ad).

**psda:**  A structure that specifies whether the service is part of the delayed start group, as specified in section [2.2.33](#Section_805b8296863d4d1e8ae8f639adf8c6cb).

**psfaf:**  A structure that specifies whether failure actions are queued when the service exits with a nonzero error code, as specified in section [2.2.41](#Section_9b244e2e82fc4c548f4fb19034faa2c4).

**pssid:**  A structure that defines the type of service SID, as specified in section [2.2.46](#Section_ea1a9acd4bb2473fae5b55969c5960fc).

**psrp:**  A structure that defines the privileges required by the service, as specified in section [2.2.48](#Section_2917121dbbe147d2ad557bce44a127e6).

**psps:**  A structure that defines the pre-shutdown settings for the service, as specified in section [2.2.45](#Section_6b55ad95df3e498fb107e77f831ba467).

**psti:**  A structure that defines the trigger settings for the service, as specified in section [2.2.54](#Section_a56f5538d13b49bfbd7c2c5b213fc0f1).[<4>](#Appendix_A_4" \o "Product behavior note 4)

**pspn:**  A structure that defines the preferred node information for the service, as specified in section [2.2.55](#Section_3767259ff0544076acd39f7ecb79cab3).[<5>](#Appendix_A_5" \o "Product behavior note 5)

### SC\_RPC\_NOTIFY\_PARAMS

The SC\_RPC\_NOTIFY\_PARAMS structure[<6>](#Appendix_A_6" \o "Product behavior note 6) contains the parameters associated with the notification information of the service status.

1. typedef struct \_SC\_RPC\_NOTIFY\_PARAMS {
2. DWORD dwInfoLevel;
3. [switch\_is(dwInfoLevel)] union {
4. [case(1)]
5. PSERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_1 pStatusChangeParam1;
6. [case(2)]
7. PSERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_2 pStatusChangeParams;
8. };
9. } SC\_RPC\_NOTIFY\_PARAMS;

**dwInfoLevel:**  A value that indicates the version of the notification structure being used.

**pStatusChangeParam1:**  A [SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_1 (section 2.2.43)](#Section_e10de8b2975e4659896bf9c5bdcc42d5) structure that contains the service status notification information.

**pStatusChangeParams:**  A [PSERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_2 (section 2.2.44)](#Section_fb661d973f844e45abbc759f19438610) structure that contains the service status notification information.

### SC\_RPC\_NOTIFY\_PARAMS\_LIST

The SC\_RPC\_NOTIFY\_PARAMS\_LIST structure[<7>](#Appendix_A_7" \o "Product behavior note 7) defines an array of service state change parameters.

1. typedef struct \_SC\_RPC\_NOTIFY\_PARAMS\_LIST {
2. BOUNDED\_DWORD\_4K cElements;
3. [size\_is(cElements)] SC\_RPC\_NOTIFY\_PARAMS NotifyParamsArray[\*];
4. } SC\_RPC\_NOTIFY\_PARAMS\_LIST,
5. \*PSC\_RPC\_NOTIFY\_PARAMS\_LIST;

**cElements:**  The number of elements in the array.

**NotifyParamsArray:**  An array of [SC\_RPC\_NOTIFY\_PARAMS (section 2.2.23)](#Section_0e88b912da92411d915ab9b7df6d64fc) structures.

### SC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSA

The SC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSA union contains information associated with the service control parameters. String values are in [**ANSI**](#gt_100cd8a6-5cb1-4895-9de6-e4a3c224a583).

1. typedef
2. [switch\_type(DWORD)]
3. union \_SC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSA {
4. [case(1)]
5. PSERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSA psrInParams;
6. } SC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSA,
7. \*PSC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSA;

**psrInParams:**  A structure that contains the service control parameter associated with a control as specified in section [2.2.30](#Section_003621fc656f41be9368324587f3904a).

### SC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSW

The SC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSW union contains information associated with the service control parameters. String values are in Unicode.

1. typedef
2. [switch\_type(DWORD)]
3. union \_SC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSW {
4. [case(1)]
5. PSERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSW psrInParams;
6. } SC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSW,
7. \*PSC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSW;

**psrInParams:**  A structure that contains the service control parameter associated with a control as specified in section [2.2.31](#Section_1ac2684aea98420cac337441af91282e).

### SC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSA

The SC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSA union contains resulting status information associated with the service control parameters. String values are in ANSI.

1. typedef
2. [switch\_type(DWORD)]
3. union \_SC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSA {
4. [case(1)]
5. PSERVICE\_CONTROL\_STATUS\_REASON\_OUT\_PARAMS psrOutParams;
6. } SC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSA,
7. \*PSC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSA;

**psrOutParams:**  A structure that contains the resulting status information associated with the service control parameter associated with a control as specified in section [2.2.32](#Section_2679fdcc5e6a40928958a625f2a4ace7).

### SC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSW

The SC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSW union contains resulting status information associated with the service control parameters. String values are in Unicode.

1. typedef
2. [switch\_type(DWORD)]
3. union \_SC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSW {
4. [case(1)]
5. PSERVICE\_CONTROL\_STATUS\_REASON\_OUT\_PARAMS psrOutParams;
6. } SC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSW,
7. \*PSC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSW;

**psrOutParams:**  A structure that contains the resulting status information associated with the service control parameter associated with a control as specified in section [2.2.32](#Section_2679fdcc5e6a40928958a625f2a4ace7).

### SC\_STATUS\_TYPE

The SC\_STATUS\_TYPE enumeration specifies the information level for the [RQueryServiceStatusEx](#Section_e27fce9dfd4547b1bdebc05a2fd53669) method.

1. typedef [v1\_enum] enum
2. {
3. SC\_STATUS\_PROCESS\_INFO = 0
4. } SC\_STATUS\_TYPE;

**SC\_STATUS\_PROCESS\_INFO:** The information level

### SERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSA

The SERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSA structure[<8>](#Appendix_A_8" \o "Product behavior note 8) contains the reason associated with the SERVICE\_CONTROL\_STOP control. String values are in ANSI.

1. typedef struct \_SERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSA {
2. DWORD dwReason;
3. [string, range(0, SC\_MAX\_COMMENT\_LENGTH)]
4. LPSTR pszComment;
5. } SERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSA,
6. \*PSERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSA;

**dwReason:**  The reason associated with the SERVICE\_CONTROL\_STOP control. This member MUST be set to a combination of one general reason code, one major reason code, and one minor reason code.



The following are the general reason codes.

| Value | Meaning |
| --- | --- |
| SERVICE\_STOP\_CUSTOM  0x20000000 | The reason code is defined by the user. If this flag is not present, the reason code is defined by the system. If this flag is specified with a system reason code, the function call fails.  Users can create custom major reason codes in the range SERVICE\_STOP\_REASON\_MAJOR\_MIN\_CUSTOM (0x00400000) through SERVICE\_STOP\_REASON\_MAJOR\_MAX\_CUSTOM (0x00ff0000) and minor reason codes in the range SERVICE\_STOP\_REASON\_MINOR\_MIN\_CUSTOM (0x00000100) through SERVICE\_STOP\_REASON\_MINOR\_MAX\_CUSTOM (0x0000FFFF). |
| SERVICE\_STOP\_PLANNED  0x40000000 | The service stop was planned. |
| SERVICE\_STOP\_UNPLANNED  0x10000000 | The service stop was not planned. |

The following are the major reason codes.

| Value | Meaning |
| --- | --- |
| SERVICE\_STOP\_REASON\_MAJOR\_APPLICATION  0x00050000 | Application issue |
| SERVICE\_STOP\_REASON\_MAJOR\_HARDWARE  0x00020000 | Hardware issue |
| SERVICE\_STOP\_REASON\_MAJOR\_NONE  0x00060000 | No major reason |
| SERVICE\_STOP\_REASON\_MAJOR\_OPERATINGSYSTEM  0x00030000 | Operating system issue |
| SERVICE\_STOP\_REASON\_MAJOR\_OTHER  0x00010000 | Other issue |
| SERVICE\_STOP\_REASON\_MAJOR\_SOFTWARE  0x00040000 | Software issue |

The following are the minor reason codes.

| Value | Meaning |
| --- | --- |
| SERVICE\_STOP\_REASON\_MINOR\_DISK  0x00000008 | Disk |
| SERVICE\_STOP\_REASON\_MINOR\_ENVIRONMENT  0x0000000a | Environment |
| SERVICE\_STOP\_REASON\_MINOR\_HARDWARE\_DRIVER  0x0000000b | Driver |
| SERVICE\_STOP\_REASON\_MINOR\_HUNG  0x00000006 | Unresponsive |
| SERVICE\_STOP\_REASON\_MINOR\_INSTALLATION  0x00000003 | Installation |
| SERVICE\_STOP\_REASON\_MINOR\_MAINTENANCE  0x00000002 | Maintenance |
| SERVICE\_STOP\_REASON\_MINOR\_MMC  0x00000016 | MMC issue |
| SERVICE\_STOP\_REASON\_MINOR\_NETWORK\_CONNECTIVITY  0x00000011 | Network connectivity |
| SERVICE\_STOP\_REASON\_MINOR\_NETWORKCARD  0x00000009 | Network card |
| SERVICE\_STOP\_REASON\_MINOR\_NONE  0x00000017 | No minor reason |
| SERVICE\_STOP\_REASON\_MINOR\_OTHER  0x00000001 | Other issue |
| SERVICE\_STOP\_REASON\_MINOR\_OTHERDRIVER  0x0000000c | Other driver event |
| SERVICE\_STOP\_REASON\_MINOR\_RECONFIG  0x00000005 | Reconfigure |
| SERVICE\_STOP\_REASON\_MINOR\_SECURITY  0x00000010 | Security issue |
| SERVICE\_STOP\_REASON\_MINOR\_SECURITYFIX  0x0000000f | Security update |
| SERVICE\_STOP\_REASON\_MINOR\_SECURITYFIX\_UNINSTALL  0x00000015 | Security update uninstall |
| SERVICE\_STOP\_REASON\_MINOR\_SERVICEPACK  0x0000000d | Service pack |
| SERVICE\_STOP\_REASON\_MINOR\_SERVICEPACK\_UNINSTALL  0x00000013 | Service pack uninstall |
| SERVICE\_STOP\_REASON\_MINOR\_SOFTWARE\_UPDATE  0x0000000e | Software update |
| SERVICE\_STOP\_REASON\_MINOR\_SOFTWARE\_UPDATE\_UNINSTALL  0x00000014 | Software update uninstall |
| SERVICE\_STOP\_REASON\_MINOR\_UNSTABLE  0x00000007 | Unstable |
| SERVICE\_STOP\_REASON\_MINOR\_UPGRADE  0x00000004 | Installation of software |
| SERVICE\_STOP\_REASON\_MINOR\_WMI  0x00000012 | WMI issue |

**pszComment:**  A pointer to a string that specifies a comment associated with the *dwReason* parameter. String values are in ANSI.

### SERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSW

The SERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSW structure[<9>](#Appendix_A_9" \o "Product behavior note 9) contains the reason associated with the SERVICE\_CONTROL\_STOP. String values are in Unicode.

1. typedef struct \_SERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSW {
2. DWORD dwReason;
3. [string, range(0, SC\_MAX\_COMMENT\_LENGTH)]
4. LPWSTR pszComment;
5. } SERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSW,
6. \*PSERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSW;

**dwReason:**  The reason associated with the SERVICE\_CONTROL\_STOP control. This member MUST be set to a combination of one general reason code, one major reason code, and one minor reason code.



The following are the general reason codes.

| Value | Meaning |
| --- | --- |
| SERVICE\_STOP\_CUSTOM  0x20000000 | The reason code is defined by the user. If this flag is not present, the reason code is defined by the system. If this flag is specified with a system reason code, the function call fails.  Users can create custom major reason codes in the range SERVICE\_STOP\_REASON\_MAJOR\_MIN\_CUSTOM (0x00400000) through SERVICE\_STOP\_REASON\_MAJOR\_MAX\_CUSTOM (0x00ff0000) and minor reason codes in the range SERVICE\_STOP\_REASON\_MINOR\_MIN\_CUSTOM (0x00000100) through SERVICE\_STOP\_REASON\_MINOR\_MAX\_CUSTOM (0x0000FFFF). |
| SERVICE\_STOP\_PLANNED  0x40000000 | The service stop was planned. |
| SERVICE\_STOP\_UNPLANNED  0x10000000 | The service stop was not planned. |

The following are the major reason codes.

| Value | Meaning |
| --- | --- |
| SERVICE\_STOP\_REASON\_MAJOR\_APPLICATION  0x00050000 | Application issue |
| SERVICE\_STOP\_REASON\_MAJOR\_HARDWARE  0x00020000 | Hardware issue |
| SERVICE\_STOP\_REASON\_MAJOR\_NONE  0x00060000 | No major reason |
| SERVICE\_STOP\_REASON\_MAJOR\_OPERATINGSYSTEM  0x00030000 | Operating system issue |
| SERVICE\_STOP\_REASON\_MAJOR\_OTHER  0x00010000 | Other issue |
| SERVICE\_STOP\_REASON\_MAJOR\_SOFTWARE  0x00040000 | Software issue |

The following are the minor reason codes.

| Value | Meaning |
| --- | --- |
| SERVICE\_STOP\_REASON\_MINOR\_DISK  0x00000008 | Disk |
| SERVICE\_STOP\_REASON\_MINOR\_ENVIRONMENT  0x0000000a | Environment |
| SERVICE\_STOP\_REASON\_MINOR\_HARDWARE\_DRIVER  0x0000000b | Driver |
| SERVICE\_STOP\_REASON\_MINOR\_HUNG  0x00000006 | Unresponsive |
| SERVICE\_STOP\_REASON\_MINOR\_INSTALLATION  0x00000003 | Installation |
| SERVICE\_STOP\_REASON\_MINOR\_MAINTENANCE  0x00000002 | Maintenance |
| SERVICE\_STOP\_REASON\_MINOR\_MMC  0x00000016 | MMC issue |
| SERVICE\_STOP\_REASON\_MINOR\_NETWORK\_CONNECTIVITY  0x00000011 | Network connectivity |
| SERVICE\_STOP\_REASON\_MINOR\_NETWORKCARD  0x00000009 | Network card |
| SERVICE\_STOP\_REASON\_MINOR\_NONE  0x00000017 | No minor reason |
| SERVICE\_STOP\_REASON\_MINOR\_OTHER  0x00000001 | Other issue |
| SERVICE\_STOP\_REASON\_MINOR\_OTHERDRIVER  0x0000000c | Other driver event |
| SERVICE\_STOP\_REASON\_MINOR\_RECONFIG  0x00000005 | Reconfigure |
| SERVICE\_STOP\_REASON\_MINOR\_SECURITY  0x00000010 | Security issue |
| SERVICE\_STOP\_REASON\_MINOR\_SECURITYFIX  0x0000000f | Security update |
| SERVICE\_STOP\_REASON\_MINOR\_SECURITYFIX\_UNINSTALL  0x00000015 | Security update uninstall |
| SERVICE\_STOP\_REASON\_MINOR\_SERVICEPACK  0x0000000d | Service pack |
| SERVICE\_STOP\_REASON\_MINOR\_SERVICEPACK\_UNINSTALL  0x00000013 | Service pack uninstall |
| SERVICE\_STOP\_REASON\_MINOR\_SOFTWARE\_UPDATE  0x0000000e | Software update |
| SERVICE\_STOP\_REASON\_MINOR\_SOFTWARE\_UPDATE\_UNINSTALL  0x00000014 | Software update uninstall |
| SERVICE\_STOP\_REASON\_MINOR\_UNSTABLE  0x00000007 | Unstable |
| SERVICE\_STOP\_REASON\_MINOR\_UPGRADE  0x00000004 | Installation of software |
| SERVICE\_STOP\_REASON\_MINOR\_WMI  0x00000012 | WMI issue |

**pszComment:**  A pointer to a string that specifies a comment associated with the *dwReason* parameter. String values are in Unicode.

### SERVICE\_CONTROL\_STATUS\_REASON\_OUT\_PARAMS

The SERVICE\_CONTROL\_STATUS\_REASON\_OUT\_PARAMS structure[<10>](#Appendix_A_10" \o "Product behavior note 10) contains the status of the service.

1. typedef struct \_SERVICE\_CONTROL\_STATUS\_REASON\_OUT\_PARAMS {
2. SERVICE\_STATUS\_PROCESS ServiceStatus;
3. } SERVICE\_CONTROL\_STATUS\_REASON\_OUT\_PARAMS,
4. \*PSERVICE\_CONTROL\_STATUS\_REASON\_OUT\_PARAMS;

**ServiceStatus:**  A [SERVICE\_STATUS\_PROCESS (section 2.2.49)](#Section_c2f0ab87eb1643718380ddf9cd29931e) structure that contains the current status of the service.

### SERVICE\_DELAYED\_AUTO\_START\_INFO

The SERVICE\_DELAYED\_AUTO\_START\_INFO structure[<11>](#Appendix_A_11" \o "Product behavior note 11) defines the delayed autostart setting of an autostart service.

1. typedef struct \_SERVICE\_DELAYED\_AUTO\_START\_INFO {
2. BOOL fDelayedAutostart;
3. } SERVICE\_DELAYED\_AUTO\_START\_INFO,
4. \*LPSERVICE\_DELAYED\_AUTO\_START\_INFO;

**fDelayedAutostart:**  A Boolean value that specifies whether to delay the start of the service. If this value is TRUE, the service is started after other autostart services are started plus a short delay of approximately two minutes. Otherwise, the service is started during the system boot. This setting is ignored unless the service is an autostart service.

If the service has other services that it is dependent on, as specified via the **lpDependencies** member of the QUERY\_SERVICE\_CONFIGA structure (section [2.2.14](#Section_daee07f590754534a6747db7fc689b36)) and the QUERY\_SERVICE\_CONFIGW structure (section [2.2.15](#Section_97200665563142ea99176f9b41f02391)), then those services are started before this service.

### SERVICE\_DESCRIPTIONA

The SERVICE\_DESCRIPTIONA structure contains the description of the service. String values are in ANSI.

1. typedef struct \_SERVICE\_DESCRIPTIONA {
2. [string, range(0, 8 \* 1024)] LPSTR lpDescription;
3. } SERVICE\_DESCRIPTIONA,
4. \*LPSERVICE\_DESCRIPTIONA;

**lpDescription:**  A pointer to a string that contains the description of the service in ANSI.

### SERVICE\_DESCRIPTIONW

The SERVICE\_DESCRIPTIONW structure contains the description of the service. String values are in Unicode.

1. typedef struct \_SERVICE\_DESCRIPTIONW {
2. [string, range(0, 8 \* 1024)] LPWSTR lpDescription;
3. } SERVICE\_DESCRIPTIONW,
4. \*LPSERVICE\_DESCRIPTIONW;

**lpDescription:**  A pointer to a string that contains the description of the service in Unicode.

### SERVICE\_DESCRIPTION\_WOW64

The SERVICE\_DESCRIPTION\_WOW64 structure defines the offset at which SERVICE\_DESRIPTIONW is present.

1. typedef struct {
2. DWORD dwDescriptionOffset;
3. } SERVICE\_DESCRIPTION\_WOW64;

**dwDescriptionOffset:**  A pointer to the offset for the [SERVICE\_DESCRIPTIONW (section 2.2.35)](#Section_3de0acf0bd9b4b0ba37602036b16a83f) structure, which contains the service description in [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8).

### SERVICE\_FAILURE\_ACTIONS\_WOW64

The SERVICE\_FAILURE\_ACTIONS\_WOW64 structure defines the action that the service controller takes on each failure of a service.

1. typedef struct {
2. DWORD dwResetPeriod;
3. DWORD dwRebootMsgOffset;
4. DWORD dwCommandOffset;
5. DWORD cActions;
6. DWORD dwsaActionsOffset;
7. } SERVICE\_FAILURE\_ACTIONS\_WOW64;

**dwResetPeriod:**  The time, in seconds, after which to reset the failure count to zero if there are no failures.

**dwRebootMsgOffset:**  The offset for the buffer containing the message that is broadcast in response to the SC\_ACTION\_REBOOT service controller action (section [2.2.18](#Section_7e3bf33ced3d4bc68781cc9fe092de9e)) to all server users prior to a server reboot.

**dwCommandOffset:**  The offset for the buffer that contains the [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) command line of the process that the process creation function executes in response to the SC\_ACTION\_RUN\_COMMAND service controller action (section 2.2.18).

**cActions:**  The number of [SC\_ACTION (section 2.2.19)](#Section_5e3e9ca16c334c94bf36c5b75262d7d6) structures in the array that is offset by the value of **dwsaActionsOffset**.

**dwsaActionsOffset:**  The offset for the buffer that contains an array of SC\_ACTION structures.

### SERVICE\_REQUIRED\_PRIVILEGES\_INFO\_WOW64

The SERVICE\_REQUIRED\_PRIVILEGES\_INFO\_WOW64 structure defines the offset at which the [SERVICE\_RPC\_REQUIRED\_PRIVILEGES\_INFO (section 2.2.48)](#Section_2917121dbbe147d2ad557bce44a127e6) structure is present.

1. typedef struct {
2. DWORD dwRequiredPrivilegesOffset;
3. } SERVICE\_REQUIRED\_PRIVILEGES\_INFO\_WOW64;

**dwRequiredPrivilegesOffset:**  Offset of the SERVICE\_RPC\_REQUIRED\_PRIVILEGES\_INFO structure.

### SERVICE\_FAILURE\_ACTIONSA

The SERVICE\_FAILURE\_ACTIONSA structure defines the action that the service controller takes on each failure of a service. String values are stored in ANSI.

1. typedef struct \_SERVICE\_FAILURE\_ACTIONSA {
2. DWORD dwResetPeriod;
3. [string, range(0, 8 \* 1024)] LPSTR lpRebootMsg;
4. [string, range(0, 8 \* 1024)] LPSTR lpCommand;
5. [range(0, 1024)] DWORD cActions;
6. [size\_is(cActions)] SC\_ACTION\* lpsaActions;
7. } SERVICE\_FAILURE\_ACTIONSA,
8. \*LPSERVICE\_FAILURE\_ACTIONSA;

**dwResetPeriod:**  The time, in seconds, after which to reset the failure count to zero if there are no failures.

**lpRebootMsg:**  The buffer that contains the message to be broadcast to server users before rebooting in response to the SC\_ACTION\_REBOOT service controller action.

**lpCommand:**  The buffer that contains the command line of the process for the process creation function to execute in response to the SC\_ACTION\_RUN\_COMMAND service controller action.

**cActions:**  The number of elements in the **lpsaActions** array.

**lpsaActions:**  A pointer to an array of [SC\_ACTION (section 2.2.19)](#Section_5e3e9ca16c334c94bf36c5b75262d7d6) structures.

The service controller counts the number of times each service has failed since the system booted. The count is reset to 0 if the service has not failed for **dwResetPeriod** seconds. When the service fails for the Nth time, the service controller performs the action specified in element [N-1] of the **lpsaActions** array. If N is greater than **cActions**, the service controller repeats the last action in the array.

### SERVICE\_FAILURE\_ACTIONSW

The SERVICE\_FAILURE\_ACTIONSW structure defines the action that the service controller takes on each failure of a service. String values are stored in Unicode.

1. typedef struct \_SERVICE\_FAILURE\_ACTIONSW {
2. DWORD dwResetPeriod;
3. [string, range(0, 8 \* 1024)] LPWSTR lpRebootMsg;
4. [string, range(0, 8 \* 1024)] LPWSTR lpCommand;
5. [range(0, 1024)] DWORD cActions;
6. [size\_is(cActions)] SC\_ACTION\* lpsaActions;
7. } SERVICE\_FAILURE\_ACTIONSW,
8. \*LPSERVICE\_FAILURE\_ACTIONSW;

**dwResetPeriod:**  The time, in seconds, after which to reset the failure count to zero if there are no failures.

**lpRebootMsg:**  The buffer that contains the message to be broadcast to server users before rebooting in response to the SC\_ACTION\_REBOOT service controller action.

**lpCommand:**  The buffer that contains the command line of the process for the process creation function to execute in response to the SC\_ACTION\_RUN\_COMMAND service controller action.

**cActions:**  The number of elements in the **lpsaActions** array.

**lpsaActions:**  A pointer to an array of [SC\_ACTION (section 2.2.19)](#Section_5e3e9ca16c334c94bf36c5b75262d7d6) structures.

The service controller counts the number of times each service has failed since the system booted. The count is reset to 0 if the service has not failed for **dwResetPeriod** seconds. When the service fails for the Nth time, the service controller performs the action specified in element [N-1] of the **lpsaActions** array. If N is greater than **cActions**, the service controller repeats the last action in the array.

### SERVICE\_FAILURE\_ACTIONS\_FLAG

The SERVICE\_FAILURE\_ACTIONS\_FLAG structure[<12>](#Appendix_A_12" \o "Product behavior note 12) defines the failure action setting of a service. This setting determines when failure actions are to be executed.

1. typedef struct \_SERVICE\_FAILURE\_ACTIONS\_FLAG {
2. BOOL fFailureActionsOnNonCrashFailures;
3. } SERVICE\_FAILURE\_ACTIONS\_FLAG,
4. \*LPSERVICE\_FAILURE\_ACTIONS\_FLAG;

**fFailureActionsOnNonCrashFailures:**  If this member is TRUE and the service has configured failure actions, the failure actions are queued if the service process terminates without reporting a status of SERVICE\_STOPPED or if it enters the SERVICE\_STOPPED state but the **dwWin32ExitCode** member of the [SERVICE\_STATUS (section 2.2.47)](#Section_4e91ff36ab5f49eda43da308e72b0b3c) structure is not ERROR\_SUCCESS.

If this member is FALSE and the service has configured failure actions, the failure actions are queued only if the service terminates without reporting a status of SERVICE\_STOPPED.

This setting is ignored unless the service has configured failure actions.

### SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS

The latest supported version of the service notification status structure.[<13>](#Appendix_A_13" \o "Product behavior note 13)

This type is declared as follows:

1. typedef SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_2 SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS, \*PSERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS;

### SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_1

The SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_1 structure defines the service status notification information. If a client uses this structure, the server copies data from this structure to the newer structure specified in [2.2.44](#Section_fb661d973f844e45abbc759f19438610), and uses the newer structure.

1. typedef struct \_SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_1 {
2. ULONGLONG ullThreadId;
3. DWORD dwNotifyMask;
4. UCHAR CallbackAddressArray[16];
5. UCHAR CallbackParamAddressArray[16];
6. SERVICE\_STATUS\_PROCESS ServiceStatus;
7. DWORD dwNotificationStatus;
8. DWORD dwSequence;
9. } SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_1,
10. \*PSERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_1;

**ullThreadId:**  Not used.

**dwNotifyMask:**  A value that specifies the status changes in which the client is interested. It MUST be one or more of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_NOTIFY\_CREATED  0x00000080 | Report when the service has been created. |
| SERVICE\_NOTIFY\_CONTINUE\_PENDING  0x00000010 | Report when the service is about to continue. |
| SERVICE\_NOTIFY\_DELETE\_PENDING  0x00000200 | Report when an application has specified the service to delete. |
| SERVICE\_NOTIFY\_DELETED  0x00000100 | Report when the service has been deleted. |
| SERVICE\_NOTIFY\_PAUSE\_PENDING  0x00000020 | Report when the service is pausing. |
| SERVICE\_NOTIFY\_PAUSED  0x00000040 | Report when the service has paused. |
| SERVICE\_NOTIFY\_RUNNING  0x00000008 | Report when the service is running. |
| SERVICE\_NOTIFY\_START\_PENDING  0x00000002 | Report when the service is starting. |
| SERVICE\_NOTIFY\_STOP\_PENDING  0x00000004 | Report when the service is stopping. |
| SERVICE\_NOTIFY\_STOPPED  0x00000001 | Report when the service has stopped. |

**CallbackAddressArray:**  Not used.

**CallbackParamAddressArray:**  Not used.

**ServiceStatus:**  A [SERVICE\_STATUS\_PROCESS (section 2.2.49)](#Section_c2f0ab87eb1643718380ddf9cd29931e) structure that contains information about the service.

**dwNotificationStatus:**  A value that indicates the notification status. If this member is ERROR\_SUCCESS, the notification has succeeded and the server adds valid information to the **ServiceStatus**, **dwNotificationTriggered**, and **pszServiceNames** members. If this member is ERROR\_REQUEST\_ABORTED or ERROR\_SERVICE\_MARKED\_FOR\_DELETE, the notification has failed.

**dwSequence:**  Not used.

### SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_2

The SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_2 structure[<14>](#Appendix_A_14" \o "Product behavior note 14) defines the service status notification information.

1. typedef struct \_SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_2 {
2. ULONGLONG ullThreadId;
3. DWORD dwNotifyMask;
4. UCHAR CallbackAddressArray[16];
5. UCHAR CallbackParamAddressArray[16];
6. SERVICE\_STATUS\_PROCESS ServiceStatus;
7. DWORD dwNotificationStatus;
8. DWORD dwSequence;
9. DWORD dwNotificationTriggered;
10. [string, range(0, 64\*1024)] PWSTR pszServiceNames;
11. } SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_2,
12. \*PSERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_2;

**ullThreadId:**  Not used.

**dwNotifyMask:**  A value that specifies the status changes in which the client is interested. It MUST be one or more of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_NOTIFY\_CREATED  0x00000080 | Report when the service has been created. |
| SERVICE\_NOTIFY\_CONTINUE\_PENDING  0x00000010 | Report when the service is about to continue. |
| SERVICE\_NOTIFY\_DELETE\_PENDING  0x00000200 | Report when an application has specified the service to delete. |
| SERVICE\_NOTIFY\_DELETED  0x00000100 | Report when the service has been deleted. |
| SERVICE\_NOTIFY\_PAUSE\_PENDING  0x00000020 | Report when the service is pausing. |
| SERVICE\_NOTIFY\_PAUSED  0x00000040 | Report when the service has paused. |
| SERVICE\_NOTIFY\_RUNNING  0x00000008 | Report when the service is running. |
| SERVICE\_NOTIFY\_START\_PENDING  0x00000002 | Report when the service is starting. |
| SERVICE\_NOTIFY\_STOP\_PENDING  0x00000004 | Report when the service is stopping. |
| SERVICE\_NOTIFY\_STOPPED  0x00000001 | Report when the service has stopped. |

**CallbackAddressArray:**  Not used.

**CallbackParamAddressArray:**  Not used.

**ServiceStatus:**  A [SERVICE\_STATUS\_PROCESS (section 2.2.49)](#Section_c2f0ab87eb1643718380ddf9cd29931e) structure that contains information about the service.

**dwNotificationStatus:** A value that indicates the notification status. If this member is ERROR\_SUCCESS, the notification has succeeded and the server adds valid information to the ServiceStatus, dwNotificationTriggered, and pszServiceNames members. If this member is ERROR\_REQUEST\_ABORTED or ERROR\_SERVICE\_MARKED\_FOR\_DELETE, the notification has failed.

**dwSequence:**  Not used.

**dwNotificationTriggered:**  The value that specifies the specific status change event that triggered the notification to the client. This MUST be one or more of the values specified in the *dwNotifyMask* parameter.

**pszServiceNames:**  A pointer to a sequence of null-terminated strings, terminated by an empty string (\0) that contains the name of the service that was created or deleted.

The forward slash, back slash, comma, and space characters are illegal in service names.

The names of the created services are prefixed by "/" to distinguish them from the names of the deleted services.

### SERVICE\_PRESHUTDOWN\_INFO

The SERVICE\_PRESHUTDOWN\_INFO structure[<15>](#Appendix_A_15" \o "Product behavior note 15) defines the time-out value in milliseconds.

1. typedef struct \_SERVICE\_PRESHUTDOWN\_INFO {
2. DWORD dwPreshutdownTimeout;
3. } SERVICE\_PRESHUTDOWN\_INFO,
4. \*LPSERVICE\_PRESHUTDOWN\_INFO;

**dwPreshutdownTimeout:**  Time, in milliseconds, that the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) waits for the service to enter the SERVICE\_STOPPED state after sending the SERVICE\_CONTROL\_PRESHUTDOWN message.

### SERVICE\_SID\_INFO

The SERVICE\_SID\_INFO structure[<16>](#Appendix_A_16" \o "Product behavior note 16) defines the type of service security identifier (SID) associated with a service.

1. typedef struct \_SERVICE\_SID\_INFO {
2. DWORD dwServiceSidType;
3. } SERVICE\_SID\_INFO,
4. \*LPSERVICE\_SID\_INFO;

**dwServiceSidType:**  The type of service SID. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_SID\_TYPE\_NONE  0x00000000 | No service SID. |
| SERVICE\_SID\_TYPE\_RESTRICTED  0x00000003 | This type includes SERVICE\_SID\_TYPE\_UNRESTRICTED. The service SID is also added to the restricted SID list of the process token. Three additional SIDs are added to the restricted SID list:  1. World SID S-1-1-0.  2. Service logon SID.  3. One [**access control entry (ACE)**](#gt_b581857f-39aa-4979-876b-daba67a40f15) that allows GENERIC\_ALL access for the service logon SID is also added to the service process token object.  If multiple services are hosted in the same process and one service has SERVICE\_SID\_TYPE\_RESTRICTED, all services MUST have SERVICE\_SID\_TYPE\_RESTRICTED. |
| SERVICE\_SID\_TYPE\_UNRESTRICTED  0x00000001 | When the service process is created, the service SID is added to the service process token with the following attributes: SE\_GROUP\_ENABLED\_BY\_DEFAULT | SE\_GROUP\_OWNER. |

### SERVICE\_STATUS

The SERVICE\_STATUS structure defines information about a service.

1. typedef struct {
2. DWORD dwServiceType;
3. DWORD dwCurrentState;
4. DWORD dwControlsAccepted;
5. DWORD dwWin32ExitCode;
6. DWORD dwServiceSpecificExitCode;
7. DWORD dwCheckPoint;
8. DWORD dwWaitHint;
9. } SERVICE\_STATUS,
10. \*LPSERVICE\_STATUS;

**dwServiceType:**  The type of service.

| Value | Meaning |
| --- | --- |
| SERVICE\_KERNEL\_DRIVER  0x00000001 | A driver service. These are services that manage devices on the system. |
| SERVICE\_FILE\_SYSTEM\_DRIVER  0x00000002 | A file system driver service. These are services that manage file systems on the system. |
| SERVICE\_WIN32\_OWN\_PROCESS  0x00000010 | A service that runs in its own process. |
| SERVICE\_WIN32\_SHARE\_PROCESS  0x00000020 | A service that shares a process with other services. |
| SERVICE\_INTERACTIVE\_PROCESS  0x00000100 | The service can interact with the desktop. |

Only SERVICE\_WIN32\_OWN\_PROCESS and SERVICE\_INTERACTIVE\_PROCESS OR SERVICE\_WIN32\_SHARE\_PROCESS and SERVICE\_INTERACTIVE\_PROCESS can be combined.

**dwCurrentState:**  The current state of the service.

| Value | Meaning |
| --- | --- |
| 0x00000005 | SERVICE\_CONTINUE\_PENDING |
| 0x00000006 | SERVICE\_PAUSE\_PENDING |
| 0x00000007 | SERVICE\_PAUSED |
| 0x00000004 | SERVICE\_RUNNING |
| 0x00000002 | SERVICE\_START\_PENDING |
| 0x00000003 | SERVICE\_STOP\_PENDING |
| 0x00000001 | SERVICE\_STOPPED |

**dwControlsAccepted:**  The control codes that the service accepts and processes in its handler function. One or more of the following values can be set. By default, all services accept the SERVICE\_CONTROL\_INTERROGATE value. A value of zero indicates that no controls are accepted.

| Value | Meaning |
| --- | --- |
| 0x00000008 | SERVICE\_ACCEPT\_PARAMCHANGE  Service can reread its startup parameters without being stopped and restarted.  This control code allows the service to receive SERVICE\_CONTROL\_PARAMCHANGE notifications. |
| 0x00000002 | SERVICE\_ACCEPT\_PAUSE\_CONTINUE  Service can be paused and continued.  This control code allows the service to receive SERVICE\_CONTROL\_PAUSE and SERVICE\_CONTROL\_CONTINUE notifications. |
| 0x00000004 | SERVICE\_ACCEPT\_SHUTDOWN  Service is notified when system shutdown occurs.  This control code enables the service to receive SERVICE\_CONTROL\_SHUTDOWN notifications from the server. |
| 0x00000001 | SERVICE\_ACCEPT\_STOP  Service can be stopped.  This control code allows the service to receive SERVICE\_CONTROL\_STOP notifications. |
| 0x00000020 | SERVICE\_ACCEPT\_HARDWAREPROFILECHANGE  Service is notified when the computer's hardware profile changes. |
| 0x00000040 | SERVICE\_ACCEPT\_POWEREVENT  Service is notified when the computer's power status changes. |
| 0x00000080 | SERVICE\_ACCEPT\_SESSIONCHANGE  Service is notified when the computer's session status changes. |
| 0x00000100 | SERVICE\_ACCEPT\_PRESHUTDOWN[<17>](#Appendix_A_17" \o "Product behavior note 17)  The service can perform preshutdown tasks.  SERVICE\_ACCEPT\_PRESHUTDOWN is sent before sending SERVICE\_CONTROL\_SHUTDOWN to give more time to services that need extra time before shutdown occurs. |
| 0x00000200 | SERVICE\_ACCEPT\_TIMECHANGE[<18>](#Appendix_A_18" \o "Product behavior note 18)  Service is notified when the system time changes. |
| 0x00000400 | SERVICE\_ACCEPT\_TRIGGEREVENT[<19>](#Appendix_A_19" \o "Product behavior note 19)  Service is notified when an event for which the service has registered occurs. |

**dwWin32ExitCode:**  An 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 sets this value to NO\_ERROR when it is running and on normal termination.

**dwServiceSpecificExitCode:**  A service-specific error code that the service returns when an error occurs while it is starting or stopping. The client SHOULD[<20>](#Appendix_A_20" \o "Product behavior note 20) ignore this value unless the **dwWin32ExitCode** member is set to ERROR\_SERVICE\_SPECIFIC\_ERROR.

**dwCheckPoint:**  A value that the service increments periodically to report its progress during a lengthy start, stop, pause, or continue operation. This value is zero when the service state is SERVICE\_PAUSED, SERVICE\_RUNNING, or SERVICE\_STOPPED.

**dwWaitHint:**  An estimate of the amount of time, in milliseconds, that the service expects a pending start, stop, pause, or continue operation to take before the service makes its next status update. Before the specified amount of time has elapsed, the service makes its next call to the SetServiceStatus function with either an incremented **dwCheckPoint** value or a change in **dwCurrentState**. If the time specified by **dwWaitHint** passes, and **dwCheckPoint** has not been incremented or **dwCurrentState** has not changed, the server can assume that an error has occurred and the service can be stopped. However, if the service shares a process with other services, the server cannot terminate the service application because it would have to terminate the other services sharing the process as well.

### SERVICE\_RPC\_REQUIRED\_PRIVILEGES\_INFO

The SERVICE\_RPC\_REQUIRED\_PRIVILEGES\_INFO structure[<21>](#Appendix_A_21" \o "Product behavior note 21) defines the required privileges for a service.

1. typedef struct \_SERVICE\_RPC\_REQUIRED\_PRIVILEGES\_INFO {
2. [range(0, 1024 \* 4)] DWORD cbRequiredPrivileges;
3. [size\_is(cbRequiredPrivileges)]
4. PBYTE pRequiredPrivileges;
5. } SERVICE\_RPC\_REQUIRED\_PRIVILEGES\_INFO,
6. \*LPSERVICE\_RPC\_REQUIRED\_PRIVILEGES\_INFO;

cbRequiredPrivileges:  Size, in bytes, of the pRequiredPrivileges buffer.

pRequiredPrivileges:  Buffer that contains the required privileges of a service in the format of a sequence of null-terminated strings, terminated by an empty string (\0). The privilege constants are detailed in [[MS-LSAD]](%5bMS-LSAD%5d.pdf#Section_1b5471ef4c334a91b079dfcbb82f05cc) section 3.1.1.2.1.

### SERVICE\_STATUS\_PROCESS

The SERVICE\_STATUS\_PROCESS structure contains information about a service that is used by the [RQueryServiceStatusEx](#Section_e27fce9dfd4547b1bdebc05a2fd53669) method.

1. typedef struct {
2. DWORD dwServiceType;
3. DWORD dwCurrentState;
4. DWORD dwControlsAccepted;
5. DWORD dwWin32ExitCode;
6. DWORD dwServiceSpecificExitCode;
7. DWORD dwCheckPoint;
8. DWORD dwWaitHint;
9. DWORD dwProcessId;
10. DWORD dwServiceFlags;
11. } SERVICE\_STATUS\_PROCESS,
12. \*LPSERVICE\_STATUS\_PROCESS;

**dwServiceType:**  The type of service. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_KERNEL\_DRIVER  0x00000001 | A driver service. These are services that manage devices on the system. |
| SERVICE\_FILE\_SYSTEM\_DRIVER  0x00000002 | A file system driver service. These are services that manage file systems on the system. |
| SERVICE\_WIN32\_OWN\_PROCESS  0x00000010 | A service that runs in its own process. |
| SERVICE\_WIN32\_SHARE\_PROCESS  0x00000020 | A service that shares a process with other services. |
| SERVICE\_INTERACTIVE\_PROCESS  0x00000100 | The service can interact with the desktop. |

Only SERVICE\_WIN32\_OWN\_PROCESS and SERVICE\_INTERACTIVE\_PROCESS or SERVICE\_WIN32\_SHARE\_PROCESS and SERVICE\_INTERACTIVE\_PROCESS can be combined.

**dwCurrentState:**  The current state of the service. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| 0x00000005 | SERVICE\_CONTINUE\_PENDING |
| 0x00000006 | SERVICE\_PAUSE\_PENDING |
| 0x00000007 | SERVICE\_PAUSED |
| 0x00000004 | SERVICE\_RUNNING |
| 0x00000002 | SERVICE\_START\_PENDING |
| 0x00000003 | SERVICE\_STOP\_PENDING |
| 0x00000001 | SERVICE\_STOPPED |

**dwControlsAccepted:**  The control codes that the service accepts and processes in its handler function. This bit mask MUST be set to zero or more of the following values. The value of dwControlsAccepted is 0x00000000 if the service type is SERVICE\_KERNEL\_DRIVER or SERVICE\_FILE\_SYSTEM\_DRIVER.

| Value | Meaning |
| --- | --- |
| 0x00000008 | SERVICE\_ACCEPT\_PARAMCHANGE  Service can reread its startup parameters without being stopped and restarted. |
| 0x00000002 | SERVICE\_ACCEPT\_PAUSE\_CONTINUE  Service can be paused and continued. |
| 0x00000004 | SERVICE\_ACCEPT\_SHUTDOWN  Service is notified when system shutdown occurs. |
| 0x00000001 | SERVICE\_ACCEPT\_STOP  Service can be stopped. |
| 0x00000020 | SERVICE\_ACCEPT\_HARDWAREPROFILECHANGE  Service is notified when the computer hardware profile changes. |
| 0x00000040 | SERVICE\_ACCEPT\_POWEREVENT  Service is notified when the computer power status changes. |
| 0x00000080 | SERVICE\_ACCEPT\_SESSIONCHANGE  Service is notified when the computer session status changes. |
| 0x00000100 | SERVICE\_ACCEPT\_PRESHUTDOWN[<22>](#Appendix_A_22" \o "Product behavior note 22)  The service can perform preshutdown tasks.  SERVICE\_ACCEPT\_PRESHUTDOWN is sent before sending SERVICE\_CONTROL\_SHUTDOWN to give more time to services that need extra time before shutdown occurs. |
| 0x00000200 | SERVICE\_ACCEPT\_TIMECHANGE[<23>](#Appendix_A_23" \o "Product behavior note 23)  Service is notified when the system time changes. |
| 0x00000400 | SERVICE\_ACCEPT\_TRIGGEREVENT[<24>](#Appendix_A_24" \o "Product behavior note 24)  Service is notified when an event for which the service has registered occurs. |

**dwWin32ExitCode:**  An error code that the service uses to report an error that occurs when it is starting or stopping.

**dwServiceSpecificExitCode:**  A service-specific error code that the service returns when an error occurs while it is starting or stopping.

**dwCheckPoint:**  A value that the service increments periodically to report its progress during a lengthy start, stop, pause, or continue operation.

**dwWaitHint:**  An estimate of the amount of time, in milliseconds, that the service expects a pending start, stop, pause, or continue operation to take before the service makes its next status update.

**dwProcessId:**  A process identifier of the service. A value of 0 indicates that the service is not started.

**dwServiceFlags:**  The bit flags that describe the process in which the service is running. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| 0x00000000 | Service is either running in a process that is not a system process, or the service is not running at all. In a nonsystem process, **dwProcessId** is nonzero. If the service is not running, **dwProcessId** is 0. |
| 0x00000001 | Service runs in a system process that MUST always be running. |

### STRING\_PTRSA

The STRING\_PTRSA structure defines a pointer to an ANSI character string.

1. typedef struct \_STRING\_PTRSA {
2. [string, range(0, SC\_MAX\_ARGUMENT\_LENGTH)]
3. LPSTR StringPtr;
4. } STRING\_PTRSA,
5. \*PSTRING\_PTRSA,
6. \*LPSTRING\_PTRSA;

**StringPtr:**  Pointer to an ANSI character string.

### STRING\_PTRSW

The STRING\_PTRSW structure defines a pointer to a Unicode character string.

1. typedef struct \_STRING\_PTRSW {
2. [string, range(0, SC\_MAX\_ARGUMENT\_LENGTH)]
3. wchar\_t\* StringPtr;
4. } STRING\_PTRSW,
5. \*PSTRING\_PTRSW,
6. \*LPSTRING\_PTRSW;

**StringPtr:**  A pointer to a Unicode character string.

### SERVICE\_TRIGGER\_SPECIFIC\_DATA\_ITEM

The SERVICE\_TRIGGER\_SPECIFIC\_DATA\_ITEM [<25>](#Appendix_A_25" \o "Product behavior note 25) structure contains information about one trigger data item of a service.

1. typedef struct \_SERVICE\_TRIGGER\_SPECIFIC\_DATA\_ITEM {
2. DWORD dwDataType;
3. [range(0, 1024)] DWORD cbData;
4. [size\_is(cbData)] PBYTE pData;
5. } SERVICE\_TRIGGER\_SPECIFIC\_DATA\_ITEM,
6. \*PSERVICE\_TRIGGER\_SPECIFIC\_DATA\_ITEM;

**dwDataType:**  The type of trigger data. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| 0x00000001 | SERVICE\_TRIGGER\_DATA\_TYPE\_BINARY |
| 0x00000002 | SERVICE\_TRIGGER\_DATA\_TYPE\_STRING |

**cbData:**  Size in bytes of the data in pData.

**pData:**  Trigger data. When **dwDataType** is set equal to 0x00000002 (SERVICE\_TRIGGER\_DATA\_TYPE\_STRING), the encoding is [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string and includes a terminating null character. This string can contain data in the format of a sequence of null-terminated strings, terminated by an empty string (\0).

### SERVICE\_TRIGGER

The SERVICE\_TRIGGER [<26>](#Appendix_A_26" \o "Product behavior note 26) structure contains information about one trigger of a service.

1. typedef struct \_SERVICE\_TRIGGER {
2. DWORD dwTriggerType;
3. DWORD dwAction;
4. GUID\* pTriggerSubtype;
5. [range(0, 64)] DWORD cDataItems;
6. [size\_is(cDataItems)] PSERVICE\_TRIGGER\_SPECIFIC\_DATA\_ITEM pDataItems;
7. } SERVICE\_TRIGGER,
8. \*PSERVICE\_TRIGGER;

**dwTriggerType:**  The type of trigger. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| 0x00000001 | SERVICE\_TRIGGER\_TYPE\_DEVICE\_INTERFACE\_ARRIVAL  The event is triggered when a device of the specified [**device interface class**](#gt_0b3a092c-d22a-46f6-9111-dfa1506a8661) arrives or is present when the system starts. This trigger event is commonly used to start a service.  Interface arrival occurs when a device belonging to a device interface class has been inserted.  The **pTriggerSubtype** member specifies the device interface class [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1), as defined in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.3.4. These GUIDs are defined in device-specific header files provided with the Windows Driver Kit (WDK) [[MSDN-WinDriverKit]](https://go.microsoft.com/fwlink/?LinkId=151330).  The **pDataItems** member specifies one or more hardware ID and compatible ID strings for the device interface class. Strings MUST be [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8). If more than one string is specified, the event is triggered if any one of the strings matches. For example, the Wpdbusenum service is started when a device of device interface class GUID\_DEVINTERFACE\_DISK {53f56307-b6bf-11d0-94f2-00a0c91efb8b} and a hardware ID string of "USBSTOR\GenDisk" arrives. |
| 0x00000002 | SERVICE\_TRIGGER\_TYPE\_IP\_ADDRESS\_AVAILABILITY  The event is triggered when the first IP address on the TCP/IP networking stack becomes available or the last IP address on the stack becomes unavailable. This trigger event can be used to start or stop a service.  The **pTriggerSubtype** member specifies NETWORK\_MANAGER\_FIRST\_IP\_ADDRESS\_ARRIVAL\_GUID or NETWORK\_MANAGER\_LAST\_IP\_ADDRESS\_REMOVAL\_GUID.  The **pDataItems** member is not used. |
| 0x00000003 | SERVICE\_TRIGGER\_TYPE\_DOMAIN\_JOIN  The event is triggered when the computer joins or leaves a domain. This trigger event can be used to start or stop a service.  The **pTriggerSubtype** member specifies DOMAIN\_JOIN\_GUID or DOMAIN\_LEAVE\_GUID.  The **pDataItems** member is not used. |
| 0x00000004 | SERVICE\_TRIGGER\_TYPE\_FIREWALL\_PORT\_EVENT  The event is triggered when a firewall port is opened or approximately 60 seconds after the firewall port is closed. This trigger event can be used to start or stop a service.  The **pTriggerSubtype** member specifies FIREWALL\_PORT\_OPEN\_GUID or FIREWALL\_PORT\_CLOSE\_GUID.  The **pDataItems** member specifies the port, the protocol, and optionally the executable path and user information (SID string or name) of the service listening on the event. The "RPC" token can be used in place of the port to specify any listening socket used by RPC. The "system" token can be used in place of the executable path to specify ports created by and listened on by the Windows kernel.  The event is triggered only if all strings match. For example, if MyService hosted inside Svchost.exe is to be trigger-started when port UDP 5001 opens, the trigger-specific data would be the Unicode representation of "5001\0UDP\0%systemroot%\system32\svchost.exe\0MyService\0\0". |
| 0x00000005 | SERVICE\_TRIGGER\_TYPE\_GROUP\_POLICY  The event is triggered when a machine policy or user policy change occurs. This trigger event is commonly used to start a service.  The **pTriggerSubtype** member specifies MACHINE\_POLICY\_PRESENT\_GUID or USER\_POLICY\_PRESENT\_GUID.  The **pDataItems** member is not used. |
| 0x00000020 | SERVICE\_TRIGGER\_TYPE\_CUSTOM  The event is a custom event generated by an Event Tracing for Windows (ETW) provider. This trigger event can be used to start or stop a service.  The **pTriggerSubtype** member specifies the event provider's GUID.  The **pDataItems** member specifies trigger-specific data defined by the provider. |

**dwAction:**  The type of action to be taken on the trigger arrival. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| 0x00000001 | SERVICE\_TRIGGER\_ACTION\_SERVICE\_START |
| 0x00000002 | SERVICE\_TRIGGER\_ACTION\_SERVICE\_STOP |

**pTriggerSubtype:**  Points to a GUID that identifies the trigger event subtype. The value of this member depends on the value of the **dwTriggerType** member.

If **dwTriggerType** is SERVICE\_TRIGGER\_TYPE\_CUSTOM, **pTriggerSubtype** is the GUID that identifies the custom event provider.

If **dwTriggerType** is SERVICE\_TRIGGER\_TYPE\_DEVICE\_INTERFACE\_ARRIVAL, **pTriggerSubtype** is the GUID that identifies the device interface class.

For other trigger event types, **pTriggerSubtype** can be one of the following values.

| Value | Meaning |
| --- | --- |
| DOMAIN\_JOIN\_GUID  1ce20aba-9851-4421-9430-1ddeb766e809 | The event is triggered when the computer joins a domain. The **dwTriggerType** member MUST be SERVICE\_TRIGGER\_TYPE\_DOMAIN\_JOIN. |
| DOMAIN\_LEAVE\_GUID  ddaf516e-58c2-4866-9574-c3b615d42ea1 | The event is triggered when the computer leaves a domain. The **dwTriggerType** member MUST be SERVICE\_TRIGGER\_TYPE\_DOMAIN\_JOIN. |
| FIREWALL\_PORT\_OPEN\_GUID  b7569e07-8421-4ee0-ad10-86915afdad09 | The event is triggered when the specified firewall port is opened. The **dwTriggerType** member MUST be SERVICE\_TRIGGER\_TYPE\_FIREWALL\_PORT\_EVENT. |
| FIREWALL\_PORT\_CLOSE\_GUID  a144ed38-8e12-4de4-9d96-e64740b1a524 | The event is triggered approximately 60 seconds after the specified firewall port is closed. The **dwTriggerType** member MUST be SERVICE\_TRIGGER\_TYPE\_FIREWALL\_PORT\_EVENT. |
| MACHINE\_POLICY\_PRESENT\_GUID  659FCAE6-5BDB-4DA9-B1FF-CA2A178D46E0 | The event is triggered when the machine policy has changed. The **dwTriggerType** member MUST be SERVICE\_TRIGGER\_TYPE\_GROUP\_POLICY. |
| NETWORK\_MANAGER\_FIRST\_IP\_ADDRESS\_ARRIVAL\_GUID  4f27f2de-14e2-430b-a549-7cd48cbc8245 | The event is triggered when the first IP address on the TCP/IP networking stack becomes available. The **dwTriggerType** member MUST be SERVICE\_TRIGGER\_TYPE\_IP\_ADDRESS\_AVAILABILITY. |
| NETWORK\_MANAGER\_LAST\_IP\_ADDRESS\_REMOVAL\_GUID  cc4ba62a-162e-4648-847a-b6bdf993e335 | The event is triggered when the last IP address on the TCP/IP networking stack becomes unavailable. The **dwTriggerType** member MUST be SERVICE\_TRIGGER\_TYPE\_IP\_ADDRESS\_AVAILABILITY. |
| USER\_POLICY\_PRESENT\_GUID  54FB46C8-F089-464C-B1FD-59D1B62C3B50 | The event is triggered when the user policy has changed. The **dwTriggerType** member MUST be SERVICE\_TRIGGER\_TYPE\_GROUP\_POLICY. |

**cDataItems:**  Number of data items in the *pDataItems* array.

**pDataItems:**  Array of [SERVICE\_TRIGGER\_SPECIFIC\_DATA\_ITEM](#Section_8c33ca40ac7d413da76871fc2c36d6bd) structures.

### SERVICE\_TRIGGER\_INFO

The SERVICE\_TRIGGER\_INFO [<27>](#Appendix_A_27" \o "Product behavior note 27) structure contains trigger information about a service.

1. typedef struct \_SERVICE\_TRIGGER\_INFO {
2. [range(0, 64)] DWORD cTriggers;
3. [size\_is(cTriggers)] PSERVICE\_TRIGGER pTriggers;
4. PBYTE pReserved;
5. } SERVICE\_TRIGGER\_INFO,
6. \*PSERVICE\_TRIGGER\_INFO;

**cTriggers:**  Number of items in the pTriggers array.

**pTriggers:**  Array of triggers each element of type SERVICE\_TRIGGER.

**pReserved:**  Reserved, MUST be NULL.

### SERVICE\_PREFERRED\_NODE\_INFO

The server MUST support initializing and executing a given service within a specified node when the server is running on a system supporting Non-Uniform Memory Access (NUMA) technology [[MSDN-NUMA]](https://go.microsoft.com/fwlink/?LinkId=151239). The SERVICE\_PREFERRED\_NODE\_INFO [<28>](#Appendix_A_28" \o "Product behavior note 28) structure defines the preferred node of a service.

1. typedef struct \_SERVICE\_PREFERRED\_NODE\_INFO {
2. USHORT usPreferredNode;
3. BOOLEAN fDelete;
4. } SERVICE\_PREFERRED\_NODE\_INFO,
5. \*LPSERVICE\_PREFERRED\_NODE\_INFO;

**usPreferredNode:**  The preferred node number.

**fDelete:**  If the preferred [**NUMA node**](#gt_ffe8ba4c-a382-4105-bd75-e0b0a0dbc4db) information of the service can be deleted, set to 1; otherwise set to 0.

### svcctl Interface Constants

The following are constants that are used by the **svcctl** interface.

| Constant/value | Description |
| --- | --- |
| MAX\_SERVICE\_NAME\_LENGTH  256 | This constant is the maximum length of a service name. It is defined as an unsigned short. The length does not include the terminating null character. |
| SC\_MAX\_ACCOUNT\_NAME\_LENGTH  2048 | This constant is the maximum size of the account name strings. It is defined as an unsigned short. The length includes the terminating null character. |
| SC\_MAX\_ARGUMENT\_LENGTH  1024 | This constant is the maximum size of the argument strings. It is defined as an unsigned short. The length includes the terminating null character. |
| SC\_MAX\_ARGUMENTS  1024 | This constant is the maximum length of the *argc* parameter of the [RStartServiceA (section 3.1.4.30)](#Section_f15fc39185764e30b15875c306e1cba2) and [RStartServiceW (section 3.1.4.19)](#Section_d9be95a2cf014bdcb30f6fe4b37ada16) [**RPCs**](#gt_8a7f6700-8311-45bc-af10-82e10accd331). It is defined as an unsigned short. |
| SC\_MAX\_COMMENT\_LENGTH  128 | This constant is the maximum size of the comment strings. It is defined as an unsigned short. The length includes the terminating null character. |
| SC\_MAX\_COMPUTER\_NAME\_LENGTH  1024 | This constant is the maximum size of the computer name strings. It is defined as an unsigned short. The length includes the terminating null character. |
| SC\_MAX\_DEPEND\_SIZE  4096 | This constant is the maximum size in bytes of the dependency strings, which describe the set of startup order dependencies for a service. It is defined as an unsigned short. The length includes two terminating null characters. |
| SC\_MAX\_NAME\_LENGTH  257 | This constant is the maximum size in bytes of the name strings. It is defined as an unsigned short. The length includes the terminating null character. |
| SC\_MAX\_PATH\_LENGTH  32768 | This constant is the maximum size of the path strings. It is defined as an unsigned short. The length includes the terminating null character. |
| SC\_MAX\_PWD\_SIZE  514 | This constant is the maximum size of the password strings. It is defined as an unsigned short. The length includes the terminating null character. |

### Common Error Codes

Unless specified explicitly, the methods in the **svcctl** interface return 0 on success and a nonzero implementation-specific value on failure in the return code of the response. All failure values MUST be treated as equivalent for protocol purposes and SHOULD be simply passed back to the invoking application.

# Protocol Details

The following sections specify details of the Service Control Manager Remote Protocol, including abstract data models, interface method syntax, and message processing rules.

The client side of this protocol is simply a pass-through. That is, no additional timers or other state is required on the client side of this protocol. Calls made by the higher-layer protocol or application are passed directly to the transport, and the results returned by the transport are passed directly back to the higher-layer protocol or application.

## Server Details

The Service Control Manager Remote Protocol server handles client requests for any of the messages specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282) and operates on [**services**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) on the server. For each of those messages, the behavior of the server is specified in section 3.1.4.

### Abstract Data Model

Services are programs that execute on a machine whose life cycle and execution properties are governed by the rules defined by the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84). The state diagram that models these rules follows.

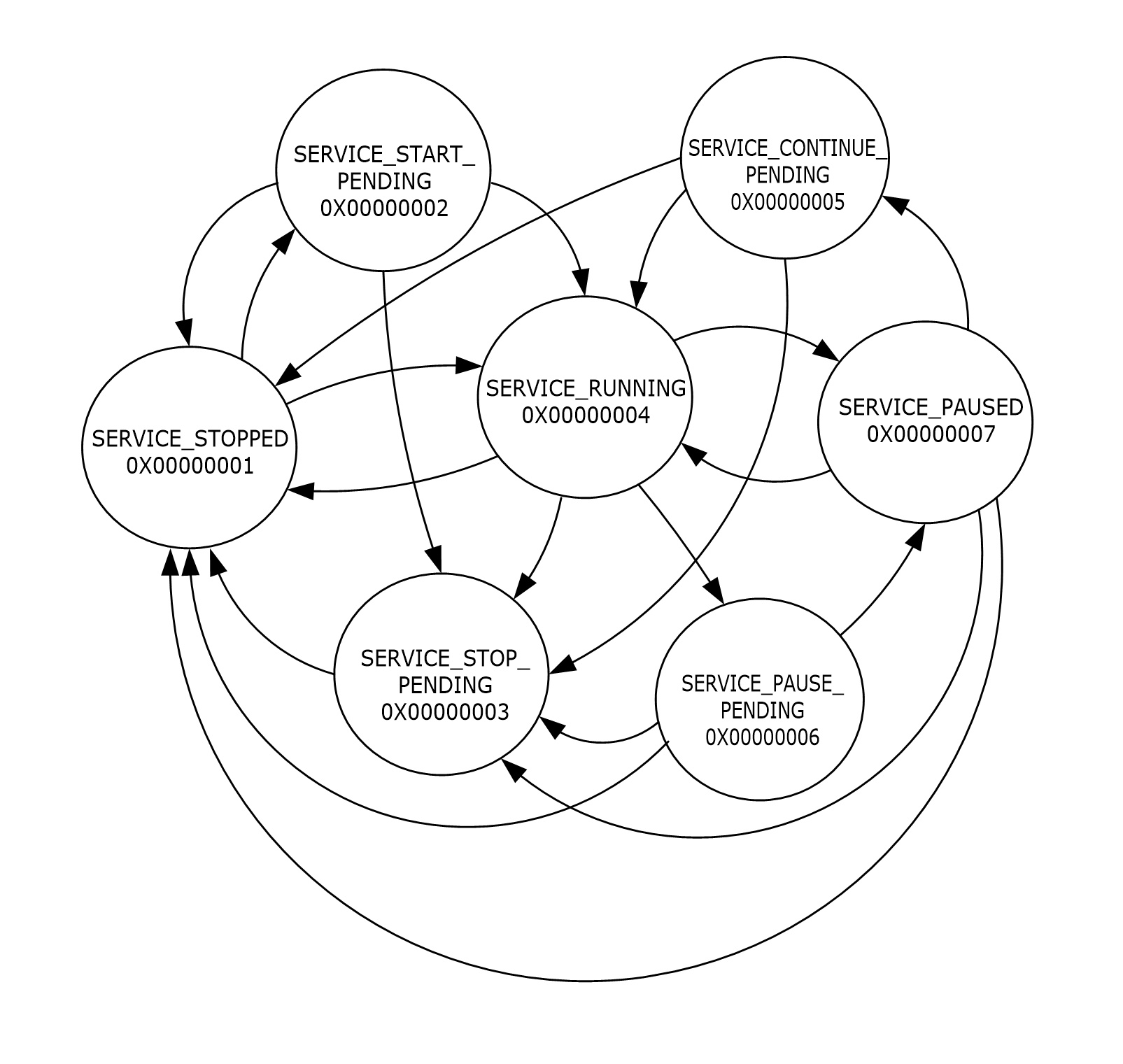


Figure 1: State Diagram in which life cycle and execution properties are governed by the rules defined in SCM

| From state | To state | Cause |
| --- | --- | --- |
| SERVICE\_STOPPED | SERVICE\_RUNNING | * The client calls the **StartService** function to start the service. For more information, see [[MSDN-STARTSERVICE]](https://go.microsoft.com/fwlink/?LinkId=90137). * The server started the [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) at system start. |
| SERVICE\_STOPPED | SERVICE\_START\_PENDING | * The client calls the **StartService** function to start the service. For more information, see [MSDN-STARTSERVICE]. * The service asks the server to change its service status to SERVICE\_START\_PENDING status using the **SetServiceStatus** function if it requires more time to initialize before it can handle requests. For more information, see [[MSDN-SetSvcStatus]](https://go.microsoft.com/fwlink/?LinkId=157504). |
| SERVICE\_START\_PENDING | SERVICE\_RUNNING | * The service asks the server to set its service status to SERVICE\_RUNNING using the **SetServiceStatus** function when it is ready to handle requests. For more information, see [MSDN-SetSvcStatus]. |
| SERVICE\_START\_PENDING | SERVICE\_STOP\_PENDING | * A client calls the **ControlService** or **ControlServiceEx** functions with SERVICE\_CONTROL\_STOP to stop the service. The server sets the service's status to SERVICE\_STOPPED. For more information, see [[MSDN-CtrlSvc]](https://go.microsoft.com/fwlink/?LinkId=157503) and [[MSDN-CtrlSvcEx]](https://go.microsoft.com/fwlink/?LinkId=157502). * The service asks the server to set its service status to SERVICE\_STOP\_PENDING using the **SetServiceStatus** function when it receives a stop request during initialization and requires time to stop. For more information, see [MSDN-SetSvcStatus]. * The server stops a service at system shutdown. |
| SERVICE\_START\_PENDING | SERVICE\_STOPPED | * A client calls the **ControlService** or **ControlServiceEx** functions with SERVICE\_CONTROL\_STOP to stop the service. The server sets the service's status to SERVICE\_STOPPED. For more information, see [MSDN-CtrlSvc] and [MSDN-CtrlSvcEx]. * The service asks the server to set its service status to SERVICE\_STOPPED using the **SetServiceStatus** function if it receives a stop request during initialization and is ready to stop. For more information, see [MSDN-SetSvcStatus]. * The server stops a service at system shutdown. |
| SERVICE\_STOP\_PENDING | SERVICE\_STOPPED | * The service asks the server to set its service status to SERVICE\_STOPPED using the **SetServiceStatus** function when it is ready to stop. For more information, see [MSDN-SetSvcStatus]. * The server stops a service at system shutdown. |
| SERVICE\_RUNNING | SERVICE\_PAUSED | * A client calls the **ControlService** or **ControlServiceEx** functions with SERVICE\_CONTROL\_PAUSE to pause the service. The server sets the service's status to SERVICE\_PAUSED. For more information, see [MSDN-CtrlSvc] and [MSDN-CtrlSvcEx]. * The service asks the server to set its service status to SERVICE\_PAUSED using the **SetServiceStatus** function if it is ready to pause. Otherwise, the service asks the server to set its service status to SERVICE\_PAUSE\_PENDING. For more information, see [MSDN-SetSvcStatus]. |
| SERVICE\_RUNNING | SERVICE\_PAUSE\_PENDING | * A client calls the **ControlService** or **ControlServiceEx** functions with SERVICE\_CONTROL\_PAUSE to pause the service. The server sets the service's status to SERVICE\_PAUSED. For more information, see [MSDN-CtrlSvc] and [MSDN-CtrlSvcEx]. * The service asks the server to set its service status to SERVICE\_PAUSE\_PENDING using the **SetServiceStatus** function if it receives a pause request and requires more time to pause. For more information, see [MSDN-SetSvcStatus]. |
| SERVICE\_RUNNING | SERVICE\_STOPPED | * A client calls the **ControlService** or **ControlServiceEx** functions with SERVICE\_CONTROL\_STOP to stop the service. The server sets the service's status to SERVICE\_STOPPED. For more information, see [MSDN-CtrlSvc] and [MSDN-CtrlSvcEx]. * The service asks the server to set its service status to SERVICE\_STOPPED using the **SetServiceStatus** function if it receives a stop request and is ready to stop. For more information, see [MSDN-SetSvcStatus]. * The server stops a service at system shutdown. |
| SERVICE\_RUNNING | SERVICE\_STOP\_PENDING | * A client calls the **ControlService** or **ControlServiceEx** functions with SERVICE\_CONTROL\_STOP to stop the service. The server sets the service's status to SERVICE\_STOPPED. For more information, see [MSDN-CtrlSvc] and [MSDN-CtrlSvcEx]. * The service asks the server to set its status to SERVICE\_STOP\_PENDING using the **SetServiceStatus** function if it receives a stop request and requires more time to stop. For more information, see [MSDN-SetSvcStatus]. * The server stops a service at system shutdown. |
| SERVICE\_PAUSE\_PENDING | SERVICE\_PAUSED | * The service asks the server to set its service status to SERVICE\_PAUSED using the **SetServiceStatus** function if it is ready to pause. For more information, see [MSDN-SetSvcStatus]. |
| SERVICE\_PAUSE\_PENDING | SERVICE\_STOP\_PENDING | * A client calls the **ControlService** or **ControlServiceEx** functions with SERVICE\_CONTROL\_STOP to stop the service. The server sets the service's status to SERVICE\_STOPPED. For more information, see [MSDN-CtrlSvc] and [MSDN-CtrlSvcEx]. * The service asks the server to set its service status to SERVICE\_STOP\_PENDING using the **SetServiceStatus** function if it receives a stop request while it is preparing to pause and requires more time to stop. For more information, see [MSDN-SetSvcStatus]. * The server stops a service at system shutdown. |
| SERVICE\_PAUSE\_PENDING | SERVICE\_STOPPED | * A client calls the **ControlService** or **ControlServiceEx** functions with SERVICE\_CONTROL\_STOP to stop the service. The server sets the service's status to SERVICE\_STOPPED. For more information, see [MSDN-CtrlSvc] and [MSDN-CtrlSvcEx]. * The service asks the server to set its service status to SERVICE\_STOPPED using the **SetServiceStatus** function when it is ready to stop. For more information, see [MSDN-SetSvcStatus]. * The server stops a service at system shutdown. |
| SERVICE\_PAUSED | SERVICE\_RUNNING | * A client calls the **ControlService** or **ControlServiceEx** functions with SERVICE\_CONTROL\_CONTINUE to resume a paused service. The server sets the service's status to SERVICE\_RUNNING. For more information, see [MSDN-CtrlSvc] and [MSDN-CtrlSvcEx]. * The service asks the server to set its service status to SERVICE\_CONTINUE\_PENDING using the **SetServiceStatus** function. For more information, see [MSDN-SetSvcStatus]. |
| SERVICE\_PAUSED | SERVICE\_CONTINUE\_PENDING | * A client calls the **ControlService** or **ControlServiceEx** functions with SERVICE\_CONTROL\_CONTINUE to resume a paused service. The server sets the service's status to SERVICE\_RUNNING. For more information, see [MSDN-CtrlSvc] and [MSDN-CtrlSvcEx]. * The service asks the server to set its service status to SERVICE\_CONTINUE\_PENDING using the **SetServiceStatus** function if it receives a continue request while it is paused and requires more time to resume. For more information, see [MSDN-SetSvcStatus]. |
| SERVICE\_PAUSED | SERVICE\_STOP\_PENDING | * A client calls the **ControlService** or **ControlServiceEx** functions with SERVICE\_CONTROL\_STOP to stop the service. The server sets the service's status to SERVICE\_STOPPED. For more information, see [MSDN-CtrlSvc] and [MSDN-CtrlSvcEx]. * The service asks the server to set its service status to SERVICE\_STOP\_PENDING using the **SetServiceStatus** function if it receives a stop request while it is paused and requires more time to stop. For more information, see [MSDN-SetSvcStatus]. * The server stops a service at system shutdown. |
| SERVICE\_PAUSED | SERVICE\_STOPPED | * A client calls the **ControlService** or **ControlServiceEx** functions with SERVICE\_CONTROL\_STOP to stop the service. The server sets the service's status to SERVICE\_STOPPED. For more information, see [MSDN-CtrlSvc] and [MSDN-CtrlSvcEx]. * The service asks the server to set its service status to SERVICE\_STOPPED using the **SetServiceStatus** function if it receives a stop request while it is paused and is ready to stop. For more information, see [MSDN-SetSvcStatus]. * The server stops a service at system shutdown. |
| SERVICE\_CONTINUE\_PENDING | SERVICE\_RUNNING | * The service asks the server to set its service status to SERVICE\_RUNNING using the **SetServiceStatus** function if it is ready to resume. For more information, see [MSDN-SetSvcStatus]. |
| SERVICE\_CONTINUE\_PENDING | SERVICE\_STOP\_PENDING | * A client calls the **ControlService** or **ControlServiceEx** functions with SERVICE\_CONTROL\_STOP to stop the service. The server sets the service's status to SERVICE\_STOPPED. For more information, see [MSDN-CtrlSvc] and [MSDN-CtrlSvcEx]. * The service asks the server to set its service status to SERVICE\_STOP\_PENDING using the **SetServiceStatus** function if it receives a stop request while it is resuming and requires more time to stop. For more information, see [MSDN-SetSvcStatus]. * The server stops a service at system shutdown. |
| SERVICE\_CONTINUE\_PENDING | SERVICE\_STOPPED | * A client calls the **ControlService** or **ControlServiceEx** functions with SERVICE\_CONTROL\_STOP to stop the service. The server sets the service's status to SERVICE\_STOPPED. For more information, see [MSDN-CtrlSvc] and [MSDN-CtrlSvcEx]. * The service asks the server to set its service status to SERVICE\_STOPPED using the **SetServiceStatus** function if it receives a stop request while it is resuming and is ready to stop. For more information, see [MSDN-SetSvcStatus]. * The server stops a service at system shutdown. |

The Service Control Manager Remote Protocol is used to manage these services on a remote machine by operating on the SCM on that machine.

The Service Control Manager maintains the following ADM elements.

| Value | Meaning |
| --- | --- |
| **SCM database** | A collection of service records. |
| **SecurityDescriptor** | A [**security descriptor**](#gt_e5213722-75a9-44e7-b026-8e4833f0d350), as specified in [[MS-AZOD]](%5bMS-AZOD%5d.pdf#Section_5a0a0a3ec7a742e1b5f2cc8d8bd9739e) section 1.1.1.3, that is used to control an access to the **SCM database**. |
| **GroupList** | An ordered list of strings that services can specify as a **ServiceGroup**. |
| **BootAccepted** | A flag indicating whether a successful call to RNotifyBootConfigStatus has already been made to the server.  This element is not accessible via any method and is internal to the protocol implementation. |

The **SCM database** is used by the Service Control Manager to add, modify, or configure services. Updates to the database are atomic. In the database there is a unique record, known as the service record, used to represent each installed service. A unique service name is used as the key for each service record.

The Service Record maintains the following ADM elements.

| Value | Meaning |
| --- | --- |
| **ServiceName** | A unique name for the service.   * Used as the key for the service record in the SCM database. * The string has a maximum length of SC\_MAX\_NAME\_LENGTH. * Null and empty strings are not permitted. * The string is null terminated. * The forward slash, back slash, comma, and space characters are illegal in service names. * The case of the characters is preserved in the SCM database; however, service name comparisons are always case insensitive. |
| **DisplayName** | Service display name.   * [**ANSI**](#gt_100cd8a6-5cb1-4895-9de6-e4a3c224a583) and [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) character sets are supported. * This string has a maximum length of SC\_MAX\_NAME\_LENGTH. * Null and empty strings are permitted. When not null, the string has to be null terminated.   The name is case-preserved in the Service Control Manager. Display name comparisons are always case-insensitive.  Can specify a localized string using the following format:[<29>](#Appendix_A_29" \o "Product behavior note 29)   * @[path\]dllname,-strID * The string with identifier strID is loaded from dllname; the path is optional.   The **DisplayName** cannot match any other **DisplayName** or another **ServiceName**. The **DisplayName** can match the **ServiceName** if it they both refer to the same service. |
| **Description** | Description of the service.   * ANSI and Unicode character sets are supported. * This string has a maximum length of 8192 characters. * Null and empty strings are permitted. When not null, the string has to be null terminated. |
| **DependOnService** | Service that starts before this service.   * ANSI and Unicode character sets are supported. * This string has a maximum length of the size of SC\_MAX\_DEPEND\_SIZE. * Null and empty strings are permitted. When not null, the string has to be double null terminated. * Multiple service names are separated by a null. * Direct or indirect circular dependencies on the same service are not allowed. |
| **ErrorControl** | Severity of the error if this service fails to start during startup. For the supported values, see dwErrorControl in section [3.1.4.11](#Section_61ea7ed0c49d4152a164b4830f16c8a4). |
| **FailureActions** | Actions that the service controller takes on each failure of the service.  These actions are queried and set using [SERVICE\_FAILURE\_ACTIONSA (section 2.2.39)](#Section_a5b207c7114e48aea1c6ef79c1fe1854) and [SERVICE\_FAILURE\_ACTIONSW (section 2.2.40)](#Section_58032b711e5c4f2e854534b0f2e8c6ad) via the [RQueryServiceConfig2A (section 3.1.4.36)](#Section_89892356ac9049d7ad99fc2ffa2a2494), [RQueryServiceConfig2W (section 3.1.4.37)](#Section_a11c38b617244e62aef459b78c8bae4e), [RChangeServiceConfig2A (section 3.1.4.34)](#Section_7c3a257bbbc640bcbe10c8b013876b2a), and [RChangeServiceConfig2W (section 3.1.4.35)](#Section_f655d914b6244ed8b55b463f17253707) server methods. |
| **ServiceGroup** | Name of the [**service group**](#gt_8ee1e5c0-3886-409f-8707-197e6232042d) the service belongs to for the purposes of load ordering. Each service can optionally specify only one group name. |
| **ImagePath** | Full qualified path to the service binary file. |
| **ObjectName** | If the service is a user-mode program, the name of the account under which the service executes. If the service is a driver, the name of the driver object that IO manager creates for the driver in the ObjectManager namespace. |
| **Password** | Password associated with the account specified in **ObjectName**. |
| **RequiredPrivileges** | Required privileges for the service. Privileges determine the type of system operations that can be performed. The privilege constants are detailed in [[MS-LSAD]](%5bMS-LSAD%5d.pdf#Section_1b5471ef4c334a91b079dfcbb82f05cc) Privilege Data Model (section 3.1.1.2.1). |
| **ServiceSidType** | Type of service security identifier (SID). |
| **FailureActionsOnNonCrashFailures** | Failure action setting of a service that determines when **FailureActions** are to be executed. |
| **DependOnGroup** | Service groups that MUST be started before this service. |
| **Start** | Defines when to start the service. |
| **Type** | Type of service. |
| **TriggerInfo** | Trigger setting of the service.[<30>](#Appendix_A_30" \o "Product behavior note 30) |
| **PreferredNode** | Preferred node setting of the service.[<31>](#Appendix_A_31" \o "Product behavior note 31) |
| **Tag** | A number that is unique within the Group. Refer to the definition of Group as defined previously in this table.  For driver services that have SERVICE\_BOOT\_START or SERVICE\_SYSTEM\_START start types [see *dwStartType* in RChangeServiceConfigW (section 3.1.4.11), RCreateServiceW (section [3.1.4.12](#Section_6a8ca92694774dd4b766692fab07227e)), RChangeServiceConfigA (section [3.1.4.22](#Section_232575f34bfc410f8fb3ca093005434e)), RCreateServiceA (section [3.1.4.23](#Section_3ab258d687b0459e8d83a2cdd8038b78)), and RCreateServiceWOW64A (section [3.1.4.41](#Section_eab9a1a3181b432087cab655c6701e85))], the server starts each service based on its Tag's position within the Group. |
| **SecurityDescriptor** | A security descriptor, as specified in [MS-AZOD] section 1.1.1.3, that describes the client access rights for changing service configuration. |
| **ServiceStatus** | The server maintains a SERVICE\_STATUS (section [2.2.47](#Section_4e91ff36ab5f49eda43da308e72b0b3c)) to keep track of the service runtime information. |
| **HandleCount** | Counter for the number of RPC context handles currently created for this service record.  This element is not accessible via any method and is internal to the protocol implementation. |
| **Deleted** | The flag that is set when the service record has been marked for deletion.  This element is not accessible via any method and is internal to the protocol implementation. |

### Timers

None.

### Initialization

The Service Control Manager Remote Protocol server is initialized by registering the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) interface and listening on the RPC [**well-known endpoint**](#gt_a7498f8e-e85d-473b-9fc9-d2fffdf71c8a), as specified in section [2.1](#Section_69bbbbfa6f2e449ea5bc47db341fe81b). The server MUST then wait for Service Control Manager Remote Protocol clients to establish a connection.

### Message Processing Events and Sequencing Rules

All Service Control Manager Remote Protocol operations begin with the client connection to the remote [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) and the client request to open the SCM database. After this database is opened, an [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) is associated with this opened database, and this handle is returned to the client. The client can then perform operations on this database; for example, enumerate a list of existing services, open existing services, or install new services using this handle.

To operate on a service, the client MUST first request that the [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) be opened. After this service is opened, an RPC context handle is associated with this opened service and this handle is returned to the client. The client can then perform operations on the service; for example, change configuration, start, or stop.

When opening the database or a service, the server MUST open it with the access rights requested by the client if the client has sufficient permissions for the requested operation.

Note that the server SHOULD not open if the client does not have sufficient access rights for the requested operation. Similarly, the server MUST fail specific operations if the database or the service was not opened with sufficient access rights.

The access rights are represented as a bit field, and in addition to the standard access rights, as specified in ACCESS\_MASK of [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2), the Service Control Manager Remote Protocol MUST support the following access rights.

| Value | Meaning |
| --- | --- |
| SERVICE\_ALL\_ACCESS  0x000F01FF | In addition to all access rights in this table, SERVICE\_ALL\_ACCESS includes Delete (DE), Read Control (RC), Write DACL (WD), and Write Owner (WO) access, as specified in ACCESS\_MASK (section 2.4.3) of [MS-DTYP]. |
| SERVICE\_CHANGE\_CONFIG  0x00000002 | Required to change the configuration of a service. |
| SERVICE\_ENUMERATE\_DEPENDENTS  0x00000008 | Required to enumerate the services installed on the server. |
| SERVICE\_INTERROGATE  0x00000080 | Required to request immediate status from the service. |
| SERVICE\_PAUSE\_CONTINUE  0x00000040 | Required to pause or continue the service. |
| SERVICE\_QUERY\_CONFIG  0x00000001 | Required to query the service configuration. |
| SERVICE\_QUERY\_STATUS  0x00000004 | Required to request the service status. |
| SERVICE\_START  0x00000010 | Required to start the service. |
| SERVICE\_STOP  0x00000020 | Required to stop the service. |
| SERVICE\_USER\_DEFINED\_CONTROL  0x00000100 | Required to specify a user-defined control code. |
| SERVICE\_SET\_STATUS  0x00008000 | Required for a service to set its status. |

Specific access types for Service Control Manager object:

| Value | Meaning |
| --- | --- |
| SC\_MANAGER\_LOCK  0x00000008 | Required to lock the SCM database. |
| SC\_MANAGER\_CREATE\_SERVICE  0x00000002 | Required for a service to be created. |
| SC\_MANAGER\_ENUMERATE\_SERVICE  0x00000004 | Required to enumerate a service. |
| SC\_MANAGER\_CONNECT  0x00000001 | Required to connect to the SCM. |
| SC\_MANAGER\_QUERY\_LOCK\_STATUS  0x00000010 | Required to query the lock status of the SCM database. |
| SC\_MANAGER\_MODIFY\_BOOT\_CONFIG  0x0020 | Required to call the [RNotifyBootConfigStatus](#Section_624e57ef772d45d2ab9903455879a424) method. |

The remainder of this section describes the server behavior for the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) methods supported by the Service Control Manager Remote Protocol. The protocol clients can invoke the RPC methods specified in this section in any order after a Service Control Manager Remote Protocol session is established with the server. The outcome of the calls depends on the parameters passed to each of those calls. Clients and servers SHOULD[<32>](#Appendix_A_32" \o "Product behavior note 32) support multiplexed connections, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 3.3.1.5.8.

Methods in RPC Opnum Order

| Method | Description |
| --- | --- |
| [RCloseServiceHandle](#Section_a2a4e17409fb4e55bad3f77c4b13245c) | Closes handles to the SCM and any other associated services.  Opnum: 0 |
| [RControlService](#Section_e1c478be117f45129b6717c20a48af97) | Receives a control code for a specific service handle, as specified by the client.  Opnum: 1 |
| [RDeleteService](#Section_6744cdb8f1624be0bb3198996b6495be) | Marks the specified service for deletion from the SCM database.  Opnum: 2 |
| [RLockServiceDatabase](#Section_ff71f732e91d41898fb9a410674c63ad) | Acquires a lock on a service database.  Opnum: 3 |
| [RQueryServiceObjectSecurity](#Section_7f339950ce7347829e104e1c5924594e) | Returns a copy of the security descriptor associated with a service.  Opnum: 4 |
| [RSetServiceObjectSecurity](#Section_ea93548f39174626bef72f3f8fa39299) | Sets the security descriptor associated with a service.  Opnum: 5 |
| [RQueryServiceStatus](#Section_cf94d915b4e140e5872ba9cb3ad09b46) | Returns the current status of the specified service.  Opnum: 6 |
| [RSetServiceStatus](#Section_df67cf3b1bae4359b6841b481d27a30c) | Updates the SCM status information for the calling service.  Opnum: 7 |
| [RUnlockServiceDatabase](#Section_3456de79525049828a30debd2ea0df92) | Releases a lock on a service database.  Opnum: 8 |
| RNotifyBootConfigStatus | Reports the boot status to the SCM.  Opnum: 9 |
| Opnum10NotUsedOnWire | Reserved for local use.  Opnum: 10 |
| [RChangeServiceConfigW](#Section_61ea7ed0c49d4152a164b4830f16c8a4) | Changes the configuration parameters of a service.  Opnum: 11 |
| [RCreateServiceW](#Section_6a8ca92694774dd4b766692fab07227e) | Creates a service and adds it to the specified SCM database.  Opnum: 12 |
| [REnumDependentServicesW](#Section_6269bea89dd34092bd3367cec685d38e) | Returns the name and status of each service that depends on the specified service.  Opnum: 13 |
| [REnumServicesStatusW](#Section_22b4ff3d29c6481fb5988ce66a46944a) | Enumerates services in the specified SCM database.  Opnum: 14 |
| [ROpenSCManagerW](#Section_dc84adb3d51d48eb820dba1c6ca5faf2) | Establishes a connection to the SCM on the specified computer and opens the specified SCM database.  Opnum: 15 |
| [ROpenServiceW](#Section_6d0a4225451b4132894d7cef7aecfd2d) | Opens a handle to an existing service.  Opnum: 16 |
| [RQueryServiceConfigW](#Section_89e2d5b119cf44ca969f38eea9fe7f3c) | Returns the configuration parameters of the specified service.  Opnum: 17 |
| [RQueryServiceLockStatusW](#Section_edb6dd274f894099aeeea84be7957ac0) | Returns the lock status of the specified SCM database.  Opnum: 18 |
| [RStartServiceW](#Section_d9be95a2cf014bdcb30f6fe4b37ada16) | Starts a specified service.  Opnum: 19 |
| [RGetServiceDisplayNameW](#Section_48b42c4a26b74d599b7a98533ebeb730) | Returns the display name of the specified service.  Opnum: 20 |
| [RGetServiceKeyNameW](#Section_ecd403dbcc754402a5fcfcda46106205) | Returns the key name of the specified service.  Opnum: 21 |
| Opnum22NotUsedOnWire | Reserved for local use.  Opnum: 22 |
| [RChangeServiceConfigA](#Section_232575f34bfc410f8fb3ca093005434e) | Changes the configuration parameters of a service.  Opnum: 23 |
| [RCreateServiceA](#Section_3ab258d687b0459e8d83a2cdd8038b78) | Creates a service object and adds it to the specified SCM database.  Opnum: 24 |
| [REnumDependentServicesA](#Section_e150f53356cf4d2a9e2d5dcccf3dbf61) | Returns the name and status of each service that depends on the specified service.  Opnum: 25 |
| [REnumServicesStatusA](#Section_7f4be088639945c89b3a6f54df618a94) | Enumerates services in the specified SCM database.  Opnum: 26 |
| [ROpenSCManagerA](#Section_440941b1c11f444882d4b2b71cc55f5a) | Opens a connection to the SCM from the client and opens the specified SCM database.  Opnum: 27 |
| [ROpenServiceA](#Section_ddd7c91fe4fb446e90530b6308c640eb) | Opens a handle to an existing service.  Opnum: 28 |
| [RQueryServiceConfigA](#Section_2119f180734740a1afaedaae8b76dd15) | Returns the configuration parameters of the specified service.  Opnum: 29 |
| [RQueryServiceLockStatusA](#Section_05ac6988644549ca84b0a5f88925076d) | Returns the lock status of the specified SCM database.  Opnum: 30 |
| [RStartServiceA](#Section_f15fc39185764e30b15875c306e1cba2) | Starts a specified service.  Opnum: 31 |
| [RGetServiceDisplayNameA](#Section_298c7656529a4b91a9c36d10a891015d) | Returns the display name of the specified service.  Opnum: 32 |
| [RGetServiceKeyNameA](#Section_0d04a3c9db344248824dec8ce05655ad) | Returns the key name of the specified service.  Opnum: 33 |
| Opnum34NotUsedOnWire | Reserved for local use.  Opnum: 34 |
| [REnumServiceGroupW](#Section_1781d75a0232453c9c41f05c96ce51da) | Returns the members of a [**service group**](#gt_8ee1e5c0-3886-409f-8707-197e6232042d).  Opnum: 35 |
| [RChangeServiceConfig2A](#Section_7c3a257bbbc640bcbe10c8b013876b2a) | Changes the optional configuration parameters of a service.  Opnum: 36 |
| [RChangeServiceConfig2W](#Section_f655d914b6244ed8b55b463f17253707) | Changes the optional configuration parameters of a service.  Opnum: 37 |
| [RQueryServiceConfig2A](#Section_89892356ac9049d7ad99fc2ffa2a2494) | Returns the optional configuration parameters of the specified service.  Opnum: 38 |
| [RQueryServiceConfig2W](#Section_a11c38b617244e62aef459b78c8bae4e) | Returns the optional configuration parameters of the specified service.  Opnum: 39 |
| [RQueryServiceStatusEx](#Section_e27fce9dfd4547b1bdebc05a2fd53669) | Returns the current status of the specified service, based on the specified information level.  Opnum: 40 |
| [REnumServicesStatusExA](#Section_f5512859cae94a109636eefeb0abd9a4) | Enumerates services in the specified SCM database, based on the specified information level.  Opnum: 41 |
| [REnumServicesStatusExW](#Section_fdb7df2e341e4dccad5fd16c2ac51466) | Enumerates services in the specified SCM database, based on the specified information level.  Opnum: 42 |
| Opnum43NotUsedOnWire | Reserved for local use.  Opnum: 43 |
| [RCreateServiceWOW64A](#Section_eab9a1a3181b432087cab655c6701e85) | Creates a 32-bit service in a 64-bit memory frame with the path to the file image automatically adjusted to point to the "%windir%\syswow64" area of the system drive. This method accepts ANSI strings, converting them to Unicode strings where required.  Opnum: 44 |
| [RCreateServiceWOW64W](#Section_b32e905dce864ce8a66ab2f85efaca12) | Creates a 32-bit service in a 64-bit memory frame with the path to the file image automatically adjusted to point to the "%windir%\syswow64" area of the system drive. This method directly supports Unicode string values.  Opnum: 45 |
| Opnum46NotUsedOnWire | Reserved for local use.  Opnum: 46 |
| [RNotifyServiceStatusChange](#Section_b4cb24471f8c4deea78e209bdacadea6) | Allows the client to receive a notification when the specified service is created or deleted or when its status changes.  Opnum: 47 |
| [RGetNotifyResults](#Section_7020f3e117b9495680dc583ea1509fe6) | Returns notification information whenever the specified status change occurs on a specified service.  Opnum: 48 |
| [RCloseNotifyHandle](#Section_65ce8393c4a14c3791da2f65e9f0df5d) | Unregisters the client from receiving future notifications from the server for specified status changes on a specified service.  Opnum: 49 |
| [RControlServiceExA](#Section_58f8ac09776e4853988976cc459d8ea5) | Receives a control code for a specific service.  Opnum: 50 |
| [RControlServiceExW](#Section_1c07ddac6ad54a6695c36271cd303fd3) | Receives a control code for a specific service.  Opnum: 51 |
| Opnum52NotUsedOnWire | Reserved for local use.  Opnum: 52 |
| Opnum53NotUsedOnWire | Reserved for local use.  Opnum: 53 |
| Opnum54NotUsedOnWire | Reserved for local use.  Opnum: 54 |
| Opnum55NotUsedOnWire | Reserved for local use.  Opnum: 55 |
| [RQueryServiceConfigEx](#Section_cbbf4eefd3e5487a919b5172d2394d98) | Returns the optional configuration parameters of the specified service.[<33>](#Appendix_A_33" \o "Product behavior note 33)  Opnum: 56 |

All methods MUST NOT throw exceptions.

Note that gaps in the [**opnum**](#gt_e127848e-c66d-427d-b3aa-9f904fa4ada7) numbering sequence represent opnums that MUST NOT[<34>](#Appendix_A_34" \o "Product behavior note 34) be used over the wire.

#### RCloseServiceHandle (Opnum 0)

The RCloseServiceHandle method is called by the client. In response, the server releases the handle to the specified [**service**](#gt_2dc07ca2-2b40-437e-a5ec-ed28ebfb116a) or the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database.

1. DWORD RCloseServiceHandle(
2. [in, out] LPSC\_RPC\_HANDLE hSCObject
3. );

**hSCObject:** An [SC\_RPC\_HANDLE](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) (section 2.2.4) data type that defines the handle to a service record or to the SCM database that MUST have been created previously using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282).

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns the following error code.

| Return value/code | Description |
| --- | --- |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 0xFFFF75FD | The operation completed successfully. Additionally, the passed handle was the last one created for the associated service record that was previously used in a successful call to the [RNotifyServiceStatusChange (section 3.1.4.43)](#Section_b4cb24471f8c4deea78e209bdacadea6) method. |
| 0xFFFF75FE | The operation completed successfully. Additionally, the passed handle was previously used in a successful call to the RNotifyServiceStatusChange method. |

In response to this request from the client, for a successful operation, the server MUST close the handle to the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) or the SCM database specified by the *hSCObject* parameter specified in the client request.

If *hSCObject* is the RPC control handle that has been created for the service record, the server MUST decrement the **HandleCount** field of the service record. If the **Deleted** field of the service record indicates that [RDeleteService](#Section_6744cdb8f1624be0bb3198996b6495be) has been successfully called with the RPC control handle created for the same service record, and **HandleCount** indicates that *hSCObject* is the last RPC control handle created for this service record, the server MUST delete the service record.

#### RControlService (Opnum 1)

The RControlService method receives a control code for a specific service handle, as specified by the client.

1. DWORD RControlService(
2. [in] SC\_RPC\_HANDLE hService,
3. [in] DWORD dwControl,
4. [out] LPSERVICE\_STATUS lpServiceStatus
5. );

**hService:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) that MUST have been created previously using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282).

**dwControl:** Requested control code. MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_CONTROL\_CONTINUE  0x00000003 | Notifies a paused service that it SHOULD resume. The SERVICE\_PAUSE\_CONTINUE access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_PAUSE\_CONTINUE bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |
| SERVICE\_CONTROL\_INTERROGATE  0x00000004 | Notifies a service that it SHOULD report its current status information to the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84). The SERVICE\_INTERROGATE access right MUST have been granted to the caller when the RPC control handle to the service record was created. |
| SERVICE\_CONTROL\_NETBINDADD  0x00000007 | Notifies a service that there is a new component for binding. The SERVICE\_PAUSE\_CONTINUE access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_NETBINDCHANGE bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |
| SERVICE\_CONTROL\_NETBINDDISABLE  0x0000000A | Notifies a network service that one of its bindings has been disabled. The SERVICE\_PAUSE\_CONTINUE access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_NETBINDCHANGE bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |
| SERVICE\_CONTROL\_NETBINDENABLE  0x00000009 | Notifies a network service that a disabled binding has been enabled. The SERVICE\_PAUSE\_CONTINUE access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_NETBINDCHANGE bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |
| SERVICE\_CONTROL\_NETBINDREMOVE  0x00000008 | Notifies a network service that a component for binding has been removed. The SERVICE\_PAUSE\_CONTINUE access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_NETBINDCHANGE bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |
| SERVICE\_CONTROL\_PARAMCHANGE  0x00000006 | Notifies a service that its startup parameters have changed. The SERVICE\_PAUSE\_CONTINUE access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_PARAMCHANGE bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |
| SERVICE\_CONTROL\_PAUSE  0x00000002 | Notifies a service that it SHOULD pause. The SERVICE\_PAUSE\_CONTINUE access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_PAUSE\_CONTINUE bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |
| SERVICE\_CONTROL\_STOP  0x00000001 | Notifies a service that it SHOULD stop. The SERVICE\_STOP access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_STOP bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |

Services can define their own codes in the range 128-255.

**lpServiceStatus:** Pointer to a [SERVICE\_STATUS (section 2.2.47)](#Section_4e91ff36ab5f49eda43da308e72b0b3c) structure that receives the latest service status information. The returned information reflects the most recent status that the service reported to the SCM.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The required access right had not been granted to the caller when the RPC context handle to the service record was created. |
| 1051  ERROR\_DEPENDENT\_SERVICES\_RUNNING | The service cannot be stopped because other running services are dependent on it. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 87  ERROR\_INVALID\_PARAMETER | The requested control code is undefined |
| 1052  ERROR\_INVALID\_SERVICE\_CONTROL | The requested control code is not valid, or it is unacceptable to the service. |
| 1053  ERROR\_SERVICE\_REQUEST\_TIMEOUT | The process for the service was started, but it did not respond within an implementation-specific time-out.[<35>](#Appendix_A_35" \o "Product behavior note 35) |
| 1061  ERROR\_SERVICE\_CANNOT\_ACCEPT\_CTRL | The requested control code cannot be sent to the service because the ServiceStatus.dwCurrentState in the service record is **SERVICE\_START\_PENDING** or **SERVICE\_STOP\_PENDING**. |
| 1062  ERROR\_SERVICE\_NOT\_ACTIVE | The service has not been started, or the ServiceStatus.dwCurrentState in the service record is **SERVICE\_STOPPED**. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation, the SCM MUST send the control specified in the *dwControl* parameter to the service created for the service record identified by the *hService* parameter of the client request if the type of the service record is SERVICE\_WIN32\_OWN\_PROCESS or SERVICE\_WIN32\_SHARE\_PROCESS.

If the type of the service record is SERVICE\_KERNEL\_DRIVER or SERVICE\_FILESYSTEM\_DRIVER, and *dwControl* parameter is not SERVICE\_CONTROL\_INTERROGATE or SERVICE\_CONTROL\_STOP, the SCM MUST fail the request with ERROR\_INVALID\_SERVICE\_CONTROL.

If the type of the service record is SERVICE\_KERNEL\_DRIVER or SERVICE\_FILESYSTEM\_DRIVER, the SCM MUST query the current status of the driver from the IO manager and set the ServiceStatus.dwCurrentState of the service record to SERVICE\_RUNNING if driver is loaded and SERVICE\_STOPPED if it is not.

If the *dwControl* is not SERVICE\_CONTROL\_INTERROGATE and type of the service record is SERVICE\_KERNEL\_DRIVER or SERVICE\_FILESYSTEM\_DRIVER and the driver is managed by the PnP subsystem, the SCM MUST fail the request with ERROR\_INVALID\_SERVICE\_CONTROL.

If the **ServiceStatus.dwControlsAccepted** field of the service record does not have a required SERVICE\_ACCEPT\_xxx bit set, the SCM MUST fail the request with ERROR\_INVALID\_SERVICE\_CONTROL.

In response to this request from the client, for a successful operation the server MUST set the ServiceStatus from the service record identified by the *hService* parameter of the request in the *lpServiceStatus* parameter.

The server SHOULD fill in the *lpServiceStatus* structure only when RControlService returns one of the following error codes: NO\_ERROR, ERROR\_INVALID\_SERVICE\_CONTROL, ERROR\_SERVICE\_CANNOT\_ACCEPT\_CTRL, ERROR\_DEPENDENT\_SERVICES\_RUNNING, or ERROR\_SERVICE\_NOT\_ACTIVE.

#### RDeleteService (Opnum 2)

The RDeleteService method marks the specified service for deletion from the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database.

1. DWORD RDeleteService(
2. [in] SC\_RPC\_HANDLE hService
3. );

**hService:** An [SC\_RPC\_HANDLE](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) (section 2.2.4) data type that defines the handle to the service record that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The DELETE access right MUST have been granted to the caller when the RPC context handle to the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) was created.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The DELETE access right had not been granted to the caller when the RPC context handle to the service record was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 1072  ERROR\_SERVICE\_MARKED\_FOR\_DELETE | The RDeleteService has already been called for the service record identified by the *hService* parameter. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

The server MUST change the Start in the service record to SERVICE\_DISABLED.

The server MUST set the **Deleted** field to TRUE in the service record to indicate that the deletion is pending.

The server MUST delete the service record when the last RPC context handle created for the service has been closed by a call to the [RCloseServiceHandle](#Section_a2a4e17409fb4e55bad3f77c4b13245c) (section 3.1.4.1) function.

#### RLockServiceDatabase (Opnum 3)

The RLockServiceDatabase method acquires a lock on an [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database.

1. DWORD RLockServiceDatabase(
2. [in] SC\_RPC\_HANDLE hSCManager,
3. [out] LPSC\_RPC\_LOCK lpLock
4. );

**hSCManager:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the SCM database created using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The caller MUST be granted the SC\_MANAGER\_LOCK access right when the RPC context handle is created.

**lpLock:** An [LPSC\_RPC\_LOCK (section 2.2.5)](#Section_697379216d314731b3f856f3bc510e5a) data type that defines the handle to the resulting database lock.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SC\_MANAGER\_LOCK access rights had not been granted to the caller when the RPC context handle was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 1055  ERROR\_SERVICE\_DATABASE\_LOCKED | The service database is locked. |

In response to this request from the client, for a successful operation, the server SHOULD lock the SCM database identified by the *hSCManager* parameter of the client request.[<36>](#Appendix_A_36" \o "Product behavior note 36)

After the database is locked, the server MUST respond with error code ERROR\_SERVICE\_DATABASE\_LOCKED (1055) for future RLockServiceDatabase, [RStartServiceW](#Section_d9be95a2cf014bdcb30f6fe4b37ada16), and [RStartServiceA](#Section_f15fc39185764e30b15875c306e1cba2) [**RPCs**](#gt_8a7f6700-8311-45bc-af10-82e10accd331). All other methods are unaffected.[<37>](#Appendix_A_37" \o "Product behavior note 37)

If the client holding the lock crashes or does not cleanly shut down, then an RPC context handle rundown callback executes on the server to release the lock. See [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 3.3.3.2.1 Connection Time-out.

#### RQueryServiceObjectSecurity (Opnum 4)

The RQueryServiceObjectSecurity method returns a copy of the SECURITY\_DESCRIPTOR structure associated with a service object.

1. DWORD RQueryServiceObjectSecurity(
2. [in] SC\_RPC\_HANDLE hService,
3. [in] SECURITY\_INFORMATION dwSecurityInformation,
4. [out, size\_is(cbBufSize)] LPBYTE lpSecurityDescriptor,
5. [in, range(0, 1024\*256)] DWORD cbBufSize,
6. [out] LPBOUNDED\_DWORD\_256K pcbBytesNeeded
7. );

**hService:** An [SC\_RPC\_HANDLE](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) (section 2.2.4) data type that defines the handle to a [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) or to the SCM database that MUST have been created previously using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282).

**dwSecurityInformation:** A [SECURITY\_INFORMATION (section 2.2.1)](#Section_deed7901ba2b45ceba66e071928bdfc1) type definition that specifies the security information being requested.

**lpSecurityDescriptor:** A pointer to a buffer that contains a copy of the SECURITY\_DESCRIPTOR structure (as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.6) for the specified service object.

**cbBufSize:** Size, in bytes, of the buffer to which the *lpSecurityDescriptor* parameter points.

**pcbBytesNeeded:** An [LPBOUNDED\_DWORD\_256K (section 2.2.9)](#Section_6c441817d7364a94a4440c1810a6c473) pointer to a variable that contains the number of bytes needed to return all the requested SECURITY\_DESCRIPTOR information if the method fails.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The required access rights had not been granted to the caller when the RPC context handle was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 122  ERROR\_INSUFFICIENT\_BUFFER | The data area passed to a system call is too small. |

The client MAY provide a combination of one or more SECURITY\_INFORMATION bit flags for *dwSecurityInformation*.

If SACL\_SECURITY\_INFORMATION is specified for the *dwSecurityInformation* parameter, then an ACCESS\_SYSTEM\_SECURITY right MUST have been granted to the caller when *hService* was created. (See AS in ACCESS\_MASK in [MS-DTYP] 2.4.3.)

If DACL\_SECURITY\_INFORMATION, LABEL\_SECURITY\_INFORMATION, OWNER\_SECURITY\_INFORMATION, or GROUP\_SECURITY\_INFORMATION is specified for the *dwSecurityInformation* parameter, then a READ\_CONTROL right MUST have been granted to the caller when *hService* was created. (See RC in ACCESS\_MASK in [MS-DTYP] 2.4.3.)

In response to this request from the client, for a successful operation the server MUST return a copy of the SECURITY\_DESCRIPTOR structure containing requested information obtained from the SecurityDescriptor for the service record or the SCM database identified by the *hService*.

The server MUST return SECURITY\_DESCRIPTOR in the buffer pointed to by the *lpSecurityDescriptor* parameter. The information returned depends on the values requested by the client in the *dwSecurityInformation* parameter.

The server MUST set the required buffer size, in bytes, in the *pcbBytesNeeded* parameter. If the buffer pointed to by *lpSecurityDescriptor* is insufficient to hold all the configuration data, the server MUST fail the call with ERROR\_INSUFFICIENT\_BUFFER (122).

The server MUST return ERROR\_INVALID\_PARAMETER (87) if **dwSecurityInformation** contains bits not defined for **SECURITY\_INFORMATION** (section 2.2.1).

#### RSetServiceObjectSecurity (Opnum 5)

The RSetServiceObjectSecurity method sets the SECURITY\_DESCRIPTOR structure associated with a service object.

1. DWORD RSetServiceObjectSecurity(
2. [in] SC\_RPC\_HANDLE hService,
3. [in] SECURITY\_INFORMATION dwSecurityInformation,
4. [in, size\_is(cbBufSize)] LPBYTE lpSecurityDescriptor,
5. [in] DWORD cbBufSize
6. );

**hService:** An [SC\_RPC\_HANDLE](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) (section 2.2.4) data type that defines the handle to a [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) or to the SCM database that MUST have been created previously using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282).

**dwSecurityInformation:** A [SECURITY\_INFORMATION (section 2.2.1)](#Section_deed7901ba2b45ceba66e071928bdfc1) type definition that specifies the security information being set.

**lpSecurityDescriptor:** A pointer to a buffer of bytes that contains the new security information.

**cbBufSize:** Size, in bytes, of the buffer pointed to by the *lpSecurityDescriptor* parameter.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The required access rights had not been granted to the caller when the RPC context handle was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 1072  ERROR\_SERVICE\_MARKED\_FOR\_DELETE | The **RDeleteService** method has been called with an RPC context handle identifying the same service record as the *hService* parameter for this call. |

The client MAY provide a combination of one or more SECURITY\_INFORMATION bit flags for *dwSecurityInformation*.

If SACL\_SECURITY\_INFORMATION is specified via *dwSecurityInformation*, then an ACCESS\_SYSTEM\_SECURITY right MUSThave been granted to the caller when *hService* was created. (See WD in ACCESS\_MASK in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) 2.4.3.

If LABEL\_SECURITY\_INFORMATION or OWNER\_SECURITY\_INFORMATION or GROUP\_SECURITY\_INFORMATION is specified via *dwSecurityInformation*, then a WRITE\_OWNER right MUST have been granted to the caller when *hService* was created. (See WO in ACCESS\_MASK in [MS-DTYP] 2.4.3.)

If DACL\_SECURITY\_INFORMATION is specified via *dwSecurityInformation*, then a WRITE\_DAC right MUST have been granted to the caller when *hService* was created. (See WD in ACCESS\_MASK in [MS-DTYP] 2.4.3.)

In response to this request from the client, for a successful operation the server MUST apply the information from the SECURITY\_DESCRIPTOR structure specified in the *lpSecurityDescriptor* parameter to the SecurityDescriptor associated with the SCM or the service record identified by the *hService* parameter of the request.

#### RQueryServiceStatus (Opnum 6)

The RQueryServiceStatus method returns the current status of the specified service.

1. DWORD RQueryServiceStatus(
2. [in] SC\_RPC\_HANDLE hService,
3. [out] LPSERVICE\_STATUS lpServiceStatus
4. );

**hService:** An [SC\_RPC\_HANDLE](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) (section 2.2.4) data type that defines the handle to the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) that MUST have been created previously using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SERVICE\_QUERY\_STATUS access right MUST have been granted to the caller when the RPC context handle was created.

**lpServiceStatus:** Pointer to a [SERVICE\_STATUS (section 2.2.47)](#Section_4e91ff36ab5f49eda43da308e72b0b3c) structure that contains the status information for the service.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SERVICE\_QUERY\_STATUS access right had not been granted to the caller when the RPC context handle was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 3  ERROR\_PATH\_NOT\_FOUND | The ImagePath of the service record identified by the *hService* parameter does not exist. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

If the type of the service record is SERVICE\_KERNEL\_DRIVER or SERVICE\_FILESYSTEM\_DRIVER, the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) queries the current status of the driver from the operating system and set the ServiceStatus.dwCurrentState of the service record to SERVICE\_RUNNING if driver is loaded and to SERVICE\_STOPPED if it is not.

In response to this request from the client, for a successful operation, the server MUST set the ServiceStatus from the service record identified by the *hService* parameter of the request in the *lpServiceStatus* parameter.

If no attempts to start the service for the service record identified by the *hService* parameter have been made since the last boot, the server MUST set the dwWin32ExitCode member of the *lpServiceStatus* parameter to 1077 ERROR\_SERVICE\_NEVER\_STARTED.

#### RSetServiceStatus (Opnum 7)

The RSetServiceStatus method updates the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) status information for the calling service.

1. DWORD RSetServiceStatus(
2. [in] SC\_RPC\_HANDLE hServiceStatus,
3. [in] LPSERVICE\_STATUS lpServiceStatus
4. );

**hServiceStatus:** An [SC\_RPC\_HANDLE](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) (section 2.2.4) data type that defines the handle to the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) that MUST have been created previously using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SERVICE\_SET\_STATUS access right MUST have been granted to the caller when the RPC context handle was created.

**lpServiceStatus:** Pointer to the [SERVICE\_STATUS](#Section_4e91ff36ab5f49eda43da308e72b0b3c) (section 2.2.47) structure that contains the latest status information for the service.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 6  ERROR\_INVALID\_HANDLE | Either the handle is no longer valid or the SERVICE\_SET\_STATUS access rights had not been granted to the caller when the RPC context handle was created. |
| 13  ERROR\_INVALID\_DATA | The data provided in the *lpServiceStatus* parameter is invalid. |

The server MUST return ERROR\_INVALID\_DATA (13) if the following conditions are not true:

* lpServiceStatus->dwCurrentState MUST be one of the following values:
  + SERVICE\_STOPPED
  + SERVICE\_START\_PENDING
  + SERVICE\_STOP\_PENDING
  + SERVICE\_RUNNING
  + SERVICE\_CONTINUE\_PENDING
  + SERVICE\_PAUSE\_PENDING
  + SERVICE\_PAUSED
* Only one of the following bits can be set if the SERVICE\_INTERACTIVE\_PROCESS bit is set in lpServiceStatus->dwServiceType:
  + SERVICE\_WIN32\_OWN\_PROCESS
  + SERVICE\_WIN32\_SHARE\_PROCESS
  + SERVICE\_WIN32
* Only one of the following bits can be set if the SERVICE\_INTERACTIVE\_PROCESS bit is not set in lpServiceStatus->dwServiceType:
  + SERVICE\_DRIVER
  + SERVICE\_WIN32
  + SERVICE\_WIN32\_OWN\_PROCESS
  + SERVICE\_WIN32\_SHARE\_PROCESS
* If any bits other than these are set in lpServiceStatus->dwControlsAccepted:
  + SERVICE\_ACCEPT\_STOP
  + SERVICE\_ACCEPT\_PAUSE\_CONTINUE
  + SERVICE\_ACCEPT\_SHUTDOWN
  + SERVICE\_ACCEPT\_PRESHUTDOWN
  + SERVICE\_ACCEPT\_PARAMCHANGE
  + SERVICE\_ACCEPT\_HARDWAREPROFILECHANGE
  + SERVICE\_ACCEPT\_NETBINDCHANGE
  + SERVICE\_ACCEPT\_POWEREVENT
  + SERVICE\_ACCEPT\_SESSIONCHANGE

In response to this request from the service, for a successful operation the server MUST update the ServiceStatus with the status specified by the service in the *lpServiceStatus* parameter in the service record identified by the *hServiceStatus* parameter of the client request.

In response to this request from the service, for a successful operation the server MUST transition the service for the service record identified by the *hService* parameter to a new state if the current value of ServiceStatus.dwCurrentState in the service record ServiceState is different from lpServiceStatus->dwCurrentState (section [3.1.1](#Section_3b30d63a095b4bc184ec72370fc1db61)).

#### RUnlockServiceDatabase (Opnum 8)

The RUnlockServiceDatabase method releases a lock on a service database.

1. DWORD RUnlockServiceDatabase(
2. [in, out] LPSC\_RPC\_LOCK Lock
3. );

**Lock:** An [LPSC\_RPC\_LOCK (section 2.2.5)](#Section_697379216d314731b3f856f3bc510e5a) data type that defines the database lock context handle created by a previous call to the [RLockServiceDatabase](#Section_ff71f732e91d41898fb9a410674c63ad) method.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns the following error code.

| Return value/code | Description |
| --- | --- |
| 1071  ERROR\_INVALID\_SERVICE\_LOCK | The specified RPC context handle is invalid. |

In response to this request from the client, for a successful operation the server MUST unlock the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database for the lock specified in the *Lock* parameter of the client request. Once the database is unlocked, the server MUST stop responding with error code ERROR\_SERVICE\_DATABASE\_LOCKED (1055) for future **RLockServiceDatabase**, **RStartServiceW**, and **RStartServiceA** RPCs until the database is locked again. All other methods are unaffected.[<38>](#Appendix_A_38" \o "Product behavior note 38)

#### RNotifyBootConfigStatus (Opnum 9)

The RNotifyBootConfigStatus method reports the boot status to the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84).

1. DWORD RNotifyBootConfigStatus(
2. [in, string, unique, range(0, SC\_MAX\_COMPUTER\_NAME\_LENGTH)]
3. SVCCTL\_HANDLEW lpMachineName,
4. [in] DWORD BootAcceptable
5. );

**lpMachineName:** An [SVCCTL\_HANDLEW](#Section_807438c1bff940b8921698e8136ed4d4) (section 2.2.3) data type that defines the handle that contains the **UNICODE** string name of the server to be notified.

**BootAcceptable:** A value that specifies whether the configuration used when booting the system is acceptable. MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| 0x00000000 < *value* | Server saves the configuration as the last-known good configuration. |
| 0x00000000 | Server immediately reboots, using the previously saved last-known good configuration. |

**Return Values:** The method returns ERROR\_SUCCESS (0x00000000) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The caller does not have the SC\_MANAGER\_MODIFY\_BOOT\_CONFIG access rights granted in the SCM Security Descriptor. |
| 1074  ERROR\_ALREADY\_RUNNING\_LKG | The system is currently running with the last-known-good configuration. |
| 1076  ERROR\_BOOT\_ALREADY\_ACCEPTED | The BootAccepted field of the SCM on the target machine indicated that a successful call to RNotifyBootConfigStatus has already been made. |

In response to this request from the client, for a successful operation the server MUST either save the current configuration as the last-known good configuration or MUST reboot the server by using the previously saved last-known good configuration based on the value specified in the *BootAcceptable* parameter of the client request.

In response to this request from the client, the server MUST set the **BootAccepted** field of the SCM to TRUE to indicate that a boot has been accepted. If the BootAccepted field of the SCM already indicates that a boot has been accepted, the server MUST fail the request with ERROR\_BOOT\_ALREADY\_ACCEPTED.

If the BootAcceptable parameter is 0x00000000, the method does not return.

#### RChangeServiceConfigW (Opnum 11)

The RChangeServiceConfigW method changes a service's configuration parameters in the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database.

1. DWORD RChangeServiceConfigW(
2. [in] SC\_RPC\_HANDLE hService,
3. [in] DWORD dwServiceType,
4. [in] DWORD dwStartType,
5. [in] DWORD dwErrorControl,
6. [in, string, unique, range(0, SC\_MAX\_PATH\_LENGTH)]
7. wchar\_t\* lpBinaryPathName,
8. [in, string, unique, range(0, SC\_MAX\_NAME\_LENGTH)]
9. wchar\_t\* lpLoadOrderGroup,
10. [in, out, unique] LPDWORD lpdwTagId,
11. [in, unique, size\_is(dwDependSize)]
12. LPBYTE lpDependencies,
13. [in, range(0, SC\_MAX\_DEPEND\_SIZE)]
14. DWORD dwDependSize,
15. [in, string, unique, range(0, SC\_MAX\_ACCOUNT\_NAME\_LENGTH)]
16. wchar\_t\* lpServiceStartName,
17. [in, unique, size\_is(dwPwSize)]
18. LPBYTE lpPassword,
19. [in, range(0, SC\_MAX\_PWD\_SIZE)]
20. DWORD dwPwSize,
21. [in, string, unique, range(0, SC\_MAX\_NAME\_LENGTH)]
22. wchar\_t\* lpDisplayName
23. );

**hService:** An [SC\_RPC\_HANDLE](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) (section 2.2.4) data type that defines the handle to the service record that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SERVICE\_CHANGE\_CONFIG access right MUST have been granted to the caller when the RPC context handle to the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) was created.

**dwServiceType:** A Type value for the service record (section [3.1.1](#Section_3b30d63a095b4bc184ec72370fc1db61)) that specifies the type of service. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_KERNEL\_DRIVER  0x00000001 | A driver service. These are services that manage devices on the system. |
| SERVICE\_FILE\_SYSTEM\_DRIVER  0x00000002 | A file system driver service. These are services that manage file systems on the system. |
| SERVICE\_WIN32\_OWN\_PROCESS  0x00000010 | Service that runs in its own process. |
| SERVICE\_WIN32\_SHARE\_PROCESS  0x00000020 | Service that shares a process with other services. |
| SERVICE\_NO\_CHANGE  0xFFFFFFFF | Service type does not change. |

The following flag can also be combined with the value passed in *dwServiceStartType*:

| Value | Meaning |
| --- | --- |
| SERVICE\_INTERACTIVE\_PROCESS  0x00000100 | The service can interact with the desktop. |

**dwStartType:** A Start value for the service record (section 3.1.1) that specifies when to start the service. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_BOOT\_START  0x00000000 | Starts the driver service when the system boots up. This value is valid only for driver services. |
| SERVICE\_SYSTEM\_START  0x00000001 | Starts the driver service when the system boots up. This value is valid only for driver services. The services marked SERVICE\_SYSTEM\_START are started after all SERVICE\_BOOT\_START services have been started. |
| SERVICE\_AUTO\_START  0x00000002 | Starts the service automatically during system startup. |
| SERVICE\_DEMAND\_START  0x00000003 | Starts the service when a client requests the SCM to start the service. |
| SERVICE\_DISABLED  0x00000004 | Service cannot be started. |
| SERVICE\_NO\_CHANGE  0xFFFFFFFF | Service start type does not change. |

**dwErrorControl:** An ErrorControl value for the service record (section 3.1.1) that specifies the severity of the error if the service fails to start and determines the action that the SCM takes. MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_ERROR\_IGNORE  0x00000000 | The SCM ignores the error and continues the startup operation. |
| SERVICE\_ERROR\_NORMAL  0x00000001 | The SCM logs the error and displays a message box, but continues the startup operation. |
| SERVICE\_ERROR\_SEVERE  0x00000002 | The SCM 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  0x00000003 | The SCM SHOULD log 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  0xFFFFFFFF | Service error control type does not change. |

**lpBinaryPathName:** An ImagePath value for the service record (section 3.1.1) as a pointer to a null-terminated **UNICODE** string name. The pointer contains the fully qualified path to the service binary file. The path MAY include arguments. If the path contains a space, it MUST be quoted so that it is correctly interpreted. For example, "d:\\my share\\myservice.exe" is specified as "\"d:\\my share\\myservice.exe\"".

**lpLoadOrderGroup:** A Group value for the service record (section 3.1.1) as a pointer to a null-terminated **UNICODE** string that names the load-ordering group of which this service is a member.

Specify NULL or an empty string if the service does not belong to a [**load-ordering group**](#gt_88c26bf0-5b6f-4423-82b3-1027ea5df0e3).

**lpdwTagId:** A Tag value for the service record (section 3.1.1) as a pointer to a variable that receives a tag value. The value is unique to the group specified in the *lpLoadOrderGroup* parameter.

**lpDependencies:** DependOnService and DependOnGroup values for the service record (section 3.1.1) as a pointer to an array of null-separated names of services or load ordering groups that MUST start before this service. The array is doubly null-terminated. Load ordering group names are prefixed with a "+" character (to distinguish them from service names). If the pointer is NULL or if it points to an empty string, the service has no dependencies. Cyclic dependency between services is not allowed. The character set is [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8). 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.

**dwDependSize:** The size, in bytes, of the string specified by the *lpDependencies* parameter.

**lpServiceStartName:** An ObjectName value for the service record (section 3.1.1) as a pointer to a null-terminated UNICODE string that specifies the name of the account under which the service runs.

**lpPassword:** A Password value for the service record (section 3.1.1) as a pointer to a null-terminated UNICODE string that contains the password of the account whose name was specified by the *lpServiceStartName* parameter.

**dwPwSize:** The size, in bytes, of the password specified by the *lpPassword* parameter.

**lpDisplayName:** A DisplayName value for the service record (section 3.1.1) as a pointer to a null-terminated UNICODE string that contains the display name that applications can use to identify the service for its users.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SERVICE\_CHANGE\_CONFIG access right had not been granted to the caller when the RPC context handle to the service record was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle specified is invalid. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 1057  ERROR\_INVALID\_SERVICE\_ACCOUNT | The user account name specified in the *lpServiceStartName* parameter does not exist. |
| 1059  ERROR\_CIRCULAR\_DEPENDENCY | A circular service dependency was specified. |
| 1078  ERROR\_DUPLICATE\_SERVICE\_NAME | The lpDisplayName matches either the ServiceName or the DisplayName of another service record in the service control manager database. |
| 1072  ERROR\_SERVICE\_MARKED\_FOR\_DELETE | The RDeleteService has been called for the service record identified by the *hService* parameter. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST use the values from the appropriate parameters of the client request to update the service record identified by the *hService* parameter in the SCM database:

* If the client passes NULL for *lpBinaryPathName*, the server MUST keep the existing ImagePath value.
* If the client passes NULL for *lpLoadOrderGroup*, the server MUST keep the existing ServiceGroup value.
* If the client passes NULL for *lpdwTagId*, the server MUST keep the existing Tag value.
* If the client passes NULL for *lpDependencies*, the server MUST keep the existing DependOnService and DependOnGroup values.
* If the client passes NULL for *lpServiceStartName*, the server MUST keep the existing ObjectName value.
* If the client passes NULL for *lpPassword*, the server MUST keep the existing Password value.
* If the client passes NULL for *lpDisplayName*, the server MUST keep the existing DisplayName value.

If the original service type is SERVICE\_WIN32\_OWN\_PROCESS or SERVICE\_WIN32\_SHARE\_PROCESS, the server MUST fail the call if *dwServiceType* is set to SERVICE\_FILE\_SYSTEM\_DRIVER or SERVICE\_KERNEL\_DRIVER.[<39>](#Appendix_A_39" \o "Product behavior note 39)

If *dwServiceType* is set to SERVICE\_WIN32\_OWN\_PROCESS or SERVICE\_WIN32\_SHARE\_PROCESS and is combined with the SERVICE\_INTERACTIVE\_PROCESS bit, and the **ObjectName** field of the service record is not equal to "LocalSystem", the server MUST fail the request with ERROR\_INVALID\_PARAMETER.

If the service has a PreferredNode setting and the client requested a change in service type other than SERVICE\_WIN32\_OWN\_PROCESS, the server MUST fail the call with ERROR\_INVALID\_PARAMETER (87).

If the service is a member of a load-order group has a start type of delayed autostart (see section [2.2.33](#Section_805b8296863d4d1e8ae8f639adf8c6cb)), then the server MUST fail the call with ERROR\_INVALID\_PARAMETER (87).

If *lpdwTagId* has a valid value and *lpLoadOrderGroup* is either NULL or an empty string, then the server MUST return ERROR\_INVALID\_PARAMETER.

For service record changes to apply to the running service, the service MUST be stopped and started back up, except in the case of *lpDisplayName*. Changes to *lpDisplayName* take effect immediately.

If *lpBinaryPathName* contains arguments, the server MUST pass these arguments to the service entry point.

#### RCreateServiceW (Opnum 12)

The RCreateServiceW method creates the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) in the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database.

1. DWORD RCreateServiceW(
2. [in] SC\_RPC\_HANDLE hSCManager,
3. [in, string, range(0, SC\_MAX\_NAME\_LENGTH)]
4. wchar\_t\* lpServiceName,
5. [in, string, unique, range(0, SC\_MAX\_NAME\_LENGTH)]
6. wchar\_t\* lpDisplayName,
7. [in] DWORD dwDesiredAccess,
8. [in] DWORD dwServiceType,
9. [in] DWORD dwStartType,
10. [in] DWORD dwErrorControl,
11. [in, string, range(0, SC\_MAX\_PATH\_LENGTH)]
12. wchar\_t\* lpBinaryPathName,
13. [in, string, unique, range(0, SC\_MAX\_NAME\_LENGTH)]
14. wchar\_t\* lpLoadOrderGroup,
15. [in, out, unique] LPDWORD lpdwTagId,
16. [in, unique, size\_is(dwDependSize)]
17. LPBYTE lpDependencies,
18. [in, range(0, SC\_MAX\_DEPEND\_SIZE)]
19. DWORD dwDependSize,
20. [in, string, unique, range(0, SC\_MAX\_ACCOUNT\_NAME\_LENGTH)]
21. wchar\_t\* lpServiceStartName,
22. [in, unique, size\_is(dwPwSize)]
23. LPBYTE lpPassword,
24. [in, range(0, SC\_MAX\_PWD\_SIZE)]
25. DWORD dwPwSize,
26. [out] LPSC\_RPC\_HANDLE lpServiceHandle
27. );

**hSCManager:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the SCM database created using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SC\_MANAGER\_CREATE\_SERVICE access right MUST have been granted to the caller when the RPC context handle was created.

**lpServiceName:** A pointer to a null-terminated **UNICODE** string that specifies the name of the service to install. This MUST not be NULL.

The forward slash, back slash, comma, and space characters are illegal in service names.

**lpDisplayName:** A pointer to a null-terminated **UNICODE** string that contains the display name by which user interface programs identify the service.

**dwDesiredAccess:** A value that specifies the access to the service. This MUST be one of the values as specified in section 3.1.4.

**dwServiceType:** A value that specifies the type of service. This MUST be one or a combination of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_KERNEL\_DRIVER  0x00000001 | A driver service. These are services that manage devices on the system. |
| SERVICE\_FILE\_SYSTEM\_DRIVER  0x00000002 | A file system driver service. These are services that manage file systems on the system. |
| SERVICE\_WIN32\_OWN\_PROCESS  0x00000010 | Service that runs in its own process. |
| SERVICE\_WIN32\_SHARE\_PROCESS  0x00000020 | Service that shares a process with other services. |
| SERVICE\_INTERACTIVE\_PROCESS  0x00000100 | The service can interact with the desktop. |

**dwStartType:** A value that specifies when to start the service. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_BOOT\_START  0x00000000 | Starts the driver service when the system boots up. This value is valid only for driver services. |
| SERVICE\_SYSTEM\_START  0x00000001 | Starts the driver service when the system boots up. This value is valid only for driver services. The services marked SERVICE\_SYSTEM\_START are started after all SERVICE\_BOOT\_START services have been started. |
| SERVICE\_AUTO\_START  0x00000002 | Starts the service automatically during system startup. |
| SERVICE\_DEMAND\_START  0x00000003 | Starts the service when a client requests the SCM to start the service. |
| SERVICE\_DISABLED  0x00000004 | Service cannot be started. |

**dwErrorControl:** A value that specifies the severity of the error if the service fails to start and determines the action that the SCM takes. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_ERROR\_IGNORE  0x00000000 | The SCM ignores the error and continues the startup operation. |
| SERVICE\_ERROR\_NORMAL  0x00000001 | The SCM logs the error, but continues the startup operation. |
| SERVICE\_ERROR\_SEVERE  0x00000002 | The SCM 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  0x00000003 | The SCM SHOULD log 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. |

**lpBinaryPathName:** A pointer to a null-terminated **UNICODE** string that contains the fully qualified path to the service binary file. The path MAY include arguments. If the path contains a space, it MUST be quoted so that it is correctly interpreted. For example, "d:\\my share\\myservice.exe" is specified as "\"d:\\my share\\myservice.exe\"".

**lpLoadOrderGroup:** A pointer to a null-terminated **UNICODE** string that names the load-ordering group of which this service is a member.

Specify NULL or an empty string if the service does not belong to a [**load-ordering group**](#gt_88c26bf0-5b6f-4423-82b3-1027ea5df0e3).

**lpdwTagId:** A pointer to a variable that receives a tag value. The value is unique to the group specified in the *lpLoadOrderGroup* parameter.

**lpDependencies:** A pointer to an array of null-separated names of services or load ordering groups that MUST start before this service. The array is doubly null-terminated. Load ordering group names are prefixed with a "+" character (to distinguish them from service names). If the pointer is NULL or if it points to an empty string, the service has no dependencies. Cyclic dependency between services is not allowed. The character set is [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8). 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.

**dwDependSize:** The size, in bytes, of the string specified by the *lpDependencies* parameter.

**lpServiceStartName:** A pointer to a null-terminated **UNICODE** string that specifies the name of the account under which the service SHOULD run.

**lpPassword:** A pointer to a null-terminated **UNICODE** string that contains the password of the account whose name was specified by the *lpServiceStartName* parameter.

**dwPwSize:** The size, in bytes, of the password specified by the *lpPassword* parameter.

**lpServiceHandle:** An LPSC\_RPC\_HANDLE (section 2.2.4) data type that defines the handle to the newly created service record.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SC\_MANAGER\_CREATE\_SERVICE access right had not been granted to the caller when the RPC context handle was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle specified is invalid. |
| 13  ERROR\_INVALID\_DATA | The data is invalid. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 123  ERROR\_INVALID\_NAME | The specified service name is invalid. |
| 1057  ERROR\_INVALID\_SERVICE\_ACCOUNT | The user account name specified in the *lpServiceStartName* parameter does not exist. |
| 1059  ERROR\_CIRCULAR\_DEPENDENCY | A circular service dependency was specified. |
| 1072  ERROR\_SERVICE\_MARKED\_FOR\_DELETE | The service record with a specified name already exists and RDeleteService has been called for it. |
| 1073  ERROR\_SERVICE\_EXISTS | The service record with the ServiceName matching the specified *lpServiceName* already exists. |
| 1078  ERROR\_DUPLICATE\_SERVICE\_NAME | The service record with the same DisplayName or the same ServiceName as the passed in *lpDisplayName* already exists in the service control manager database. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST use the service name specified in the *lpServiceName* parameter to create a new service record in the SCM database and use the values from the appropriate parameters of the client request to update the attributes of this newly created service record.

The server MUST treat the lpPassword as a clear-text password if the client is using RPC over TCP, ncacn\_ip\_tcp (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)). See section [2.1.2](#Section_1901c4c33d6e426989cc42e21f0e9892).

The server MUST treat the lpPassword as encrypted and decrypt it, if the client is using a RPC over NP, ncacn\_np (as specified in [MS-RPCE]). The server MUST first retrieve a [**session key**](#gt_4f67a585-fb00-4166-93e8-cf4abca8226d) as specified in [[MS-CIFS]](%5bMS-CIFS%5d.pdf#Section_d416ff7cc536406ea9514f04b2fd1d2b) (section 3.5.4.4). An [**RPC server**](#gt_ae65dac0-cd24-4e83-a946-6d1097b71553) application requests the session key of a client and then uses the routine as specified in [[MS-LSAD]](%5bMS-LSAD%5d.pdf#Section_1b5471ef4c334a91b079dfcbb82f05cc) (section 5.1.2) to decrypt the password.

If the service is created successfully, the server MUST return a handle to the service in the *lpServiceHandle* parameter with the access rights associated with this handle as specified in the *dwDesiredAccess* parameter of the client request.

If the service is created successfully, the server MUST increment the **HandleCount** field of the service record.

The only valid combinations of values for *dwServiceType* are SERVICE\_INTERACTIVE\_PROCESS and SERVICE\_WIN32\_OWN\_PROCESS or SERVICE\_INTERACTIVE\_PROCESS and SERVICE\_WIN32\_SHARE\_PROCESS. If the value of *dwServiceType* has more than one bit set and the combination of bits is not equal to SERVICE\_INTERACTIVE\_PROCESS and SERVICE\_WIN32\_OWN\_PROCESS or SERVICE\_INTERACTIVE\_PROCESS and SERVICE\_WIN32\_SHARE\_PROCESS, the server MUST fail the method and return the error ERROR\_INVALID\_PARAMETER.

If *lpBinaryPathName* contains arguments, the server MUST pass these arguments to the service entry point.

*lpdwTagId* tags MUST be evaluated by the server for driver services that have SERVICE\_BOOT\_START or SERVICE\_BOOT\_SYSTEM\_START start types.

#### REnumDependentServicesW (Opnum 13)

The REnumDependentServicesW method returns the ServiceName, DisplayName, and ServiceStatus values of service records that are listed as dependents of a specified service.

1. DWORD REnumDependentServicesW(
2. [in] SC\_RPC\_HANDLE hService,
3. [in] DWORD dwServiceState,
4. [out, size\_is(cbBufSize)] LPBYTE lpServices,
5. [in, range(0, 1024\*256)] DWORD cbBufSize,
6. [out] LPBOUNDED\_DWORD\_256K pcbBytesNeeded,
7. [out] LPBOUNDED\_DWORD\_256K lpServicesReturned
8. );

**hService:** An [SC\_RPC\_HANDLE](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SERVICE\_ENUMERATE\_DEPENDENT access right MUST have been granted to the caller when the RPC context handle to the service record was created.

**dwServiceState:** A value that specifies the service records to enumerate based on the value of their ServiceStatus.dwCurrentState. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_ACTIVE  0x00000001 | Enumerates service records that have a ServiceStatus.dwCurrentState equal to one of the following: SERVICE\_START\_PENDING, SERVICE\_STOP\_PENDING, SERVICE\_RUNNING, SERVICE\_CONTINUE\_PENDING, SERVICE\_PAUSE\_PENDING, and SERVICE\_PAUSED. |
| SERVICE\_INACTIVE  0x00000002 | Enumerates service records that have a ServiceStatus.dwCurrentState equal to SERVICE\_STOPPED. |
| SERVICE\_STATE\_ALL  0x00000003 | Enumerates service records that have a ServiceStatus.dwCurrentState equal to one of the following: SERVICE\_START\_PENDING, SERVICE\_STOP\_PENDING, SERVICE\_RUNNING, SERVICE\_CONTINUE\_PENDING, SERVICE\_PAUSE\_PENDING, SERVICE\_PAUSED, and SERVICE\_STOPPED. |

**lpServices:** A pointer to an array of [ENUM\_SERVICE\_STATUSW (section 2.2.11)](#Section_79f17326c1384fbc81705c11c4b3ca79) structures that contain the name and service status information for each dependent service in the database.

**cbBufSize:** The size, in bytes, of the array pointed to by *lpServices*.

**pcbBytesNeeded:** An [LPBOUNDED\_DWORD\_256K (section 2.2.9)](#Section_6c441817d7364a94a4440c1810a6c473) pointer to a variable that contains the number of bytes needed to store the array of service entries.

**lpServicesReturned:** An LPBOUNDED\_DWORD\_256K (section 2.2.9) pointer to a variable that contains the number of service entries returned.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SERVICE\_ENUMERATE\_DEPENDENT access right had not been granted to the caller when the RPC context handle to the service record was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 234  ERROR\_MORE\_DATA | More data is available. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST determine the list of service records that depend on the service record identified by the *hService* parameter of the client request. The server MUST return this list by setting the ServiceName, DisplayName, and ServiceStatus.dwCurrentState of each service record in this list in the array of ENUM\_SERVICE\_STATUSW (section 2.2.11) structures pointed to by the *lpServices* parameter and MUST set the number of services returned in the lpServicesReturned parameter.

If the size of the lpServices array is insufficient for the list of services returned, the server MUST fail the call with ERROR\_MORE\_DATA (234) and return the size in bytes required in the pcbBytesNeeded parameter. If the size is sufficient for data returned, the server also returns the required size, in bytes.

If the size of the lpServices array is sufficient for the list of services returned, the enumerated data MAY be in the buffer in a non-contiguous manner, and portions of the *lpServices* array MAY be empty (filled with 0x00).

The server MUST use the process described in section [3.1.7](#Section_f9ddeebeb05e423c944084fd3d14bee9), "Conversion Between ANSI and Unicode String Formats", to convert a string to the appropriate format.

The server MUST return the services in reverse sequence of the start order of the services.

The server MUST return ERROR\_INVALID\_PARAMETER (87) if a bitmask specified in *dwServiceState* contains undefined values.

#### REnumServicesStatusW (Opnum 14)

The REnumServicesStatusW method enumerates [**service records**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) in the specified [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database.

1. DWORD REnumServicesStatusW(
2. [in] SC\_RPC\_HANDLE hSCManager,
3. [in] DWORD dwServiceType,
4. [in] DWORD dwServiceState,
5. [out, size\_is(cbBufSize)] LPBYTE lpBuffer,
6. [in, range(0, 1024 \* 256)] DWORD cbBufSize,
7. [out] LPBOUNDED\_DWORD\_256K pcbBytesNeeded,
8. [out] LPBOUNDED\_DWORD\_256K lpServicesReturned,
9. [in, out, unique] LPBOUNDED\_DWORD\_256K lpResumeIndex
10. );

**hSCManager:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the SCM database that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SC\_MANAGER\_ENUMERATE\_SERVICE access right MUST have been granted to the caller when the RPC context handle to the service record was created.

**dwServiceType:** A value that specifies what types of service records to enumerate. This MUST be one or a combination of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_KERNEL\_DRIVER  0x00000001 | A driver service. These are services that manage devices on the system |
| SERVICE\_FILE\_SYSTEM\_DRIVER  0x00000002 | A file system driver service. These are services that manage file systems on the system. |
| SERVICE\_WIN32\_OWN\_PROCESS  0x00000010 | Service that runs in its own process. |
| SERVICE\_WIN32\_SHARE\_PROCESS  0x00000020 | Service that shares a process with other services. |

**dwServiceState:** A value that specifies the service records to enumerate based on the value of their **ServiceStatus.dwCurrentState**. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_ACTIVE  0x00000001 | Enumerates service records that have **ServiceStatus.dwCurrentState** equal to one of the following: SERVICE\_START\_PENDING, SERVICE\_STOP\_PENDING, SERVICE\_RUNNING, SERVICE\_CONTINUE\_PENDING, SERVICE\_PAUSE\_PENDING, and SERVICE\_PAUSED. |
| SERVICE\_INACTIVE  0x00000002 | Enumerates service records that have **ServiceStatus.dwCurrentState** equal to SERVICE\_STOPPED. |
| SERVICE\_STATE\_ALL  0x00000003 | Enumerates service records that have **ServiceStatus.dwCurrentState** equal to one of the following: SERVICE\_START\_PENDING, SERVICE\_STOP\_PENDING, SERVICE\_RUNNING, SERVICE\_CONTINUE\_PENDING, SERVICE\_PAUSE\_PENDING, SERVICE\_PAUSED, and SERVICE\_STOPPED. |

**lpBuffer:** A pointer to an array of [ENUM\_SERVICE\_STATUSW (section 2.2.11)](#Section_79f17326c1384fbc81705c11c4b3ca79) structures that contain the name and service status information for each service in the database.

**cbBufSize:** The size, in bytes, of the array pointed to by the *lpBuffer* parameter.

**pcbBytesNeeded:** An [LPBOUNDED\_DWORD\_256K (section 2.2.9)](#Section_6c441817d7364a94a4440c1810a6c473) pointer to a variable that contains the number of bytes needed to store the array of service entries.

**lpServicesReturned:** An LPBOUNDED\_DWORD\_256K (section 2.2.9) pointer to a variable that contains the number of service entries returned.

**lpResumeIndex:** An LPBOUNDED\_DWORD\_256K (section 2.2.9) pointer to a variable that specifies the current position in the status enumeration. The server MUST assign a unique number to each service for the boot session, in increasing order, and increment that number by one for each service addition. The value of the *lpResumeIndex* parameter is one of these numbers, which the server can use to determine the resumption point for the enumeration.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SM\_MANAGER\_ENUMERATE\_SERVICE access right had not been granted to the caller when the RPC context handle to the service record was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 234  ERROR\_MORE\_DATA | More data is available. |

In response to this request from the client, for a successful operation the server MUST determine the list of service records in the SCM database identified by the *hSCManager* parameter with the current value of the ServiceStatus.dwCurrentState equal to the state specified by *dwServiceState* parameter and Type equal to the *dwServiceType* parameter of the client request. The server MUST return this list by setting the ServiceName, DisplayName, and ServiceStatus of each service in this list in the array of ENUM\_SERVICE\_STATUSW (section 2.2.11) structures pointed to by the *lpBuffer* parameter and MUST set the number of services returned in the *lpServicesReturned* parameter.

If the *lpResumeIndex* value is not zero, the server MUST use that as the offset to the list of services and return only services starting at this offset. If the lpResumeIndex value is zero, the server MUST return all services. The server MUST set this parameter to zero if the operation is successful. If the *lpResumeIndex* value is set by the client to any nonzero number not returned by the server, the behavior is not defined.

If the size of the *lpBuffer* array is insufficient for the list of service records returned, the server MUST fail the call with ERROR\_MORE\_DATA (234) and return the size in bytes required in the *pcbBytesNeeded* parameter. If the size is sufficient for data returned, the server also returns the required size, in bytes. The required size is dependent on the actual number of matching service records on the system.

If the size of the *lpBuffer* array is sufficient for the list of service records returned, the enumerated data MAY be in the buffer in a non-contiguous manner, and portions of the *lpBuffer* array MAY be empty (filled with 0x00).

The server MUST return ERROR\_INVALID\_PARAMETER (87) if a bitmask specified in the *dwServiceState* parameter is zero or contains undefined values.

The server MUST return ERROR\_INVALID\_PARAMETER (87) if a bitmask specified in the *dwServiceType* parameter is zero or contains undefined values.

#### ROpenSCManagerW (Opnum 15)

The ROpenSCManagerW method establishes a connection to server and opens the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database on the specified server.

1. DWORD ROpenSCManagerW(
2. [in, string, unique, range(0, SC\_MAX\_COMPUTER\_NAME\_LENGTH)]
3. SVCCTL\_HANDLEW lpMachineName,
4. [in, string, unique, range(0, SC\_MAX\_NAME\_LENGTH)]
5. wchar\_t\* lpDatabaseName,
6. [in] DWORD dwDesiredAccess,
7. [out] LPSC\_RPC\_HANDLE lpScHandle
8. );

**lpMachineName:** An [SVCCTL\_HANDLEW](#Section_807438c1bff940b8921698e8136ed4d4) (section 2.2.3) data type that defines the pointer to a null-terminated **UNICODE** string that specifies the server's machine name.

**lpDatabaseName:** A pointer to a null-terminated **UNICODE** string that specifies the name of the SCM database to open. The parameter MUST be set to NULL, "ServicesActive", or "ServicesFailed".

**dwDesiredAccess:** A value that specifies the access to the database. This MUST be one of the values as specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282).

The client MUST also have the SC\_MANAGER\_CONNECT access right.

**lpScHandle:** An [LPSC\_RPC\_HANDLE](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the newly opened SCM database.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The client does not have the required access rights to open the SCM database on the server or the desired access is not granted to it in the SCM SecurityDescriptor. |
| 123  ERROR\_INVALID\_NAME | The specified service name is invalid. |
| 1065  ERROR\_DATABASE\_DOES\_NOT\_EXIST | The database specified does not exist. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST create an RPC context handle to the SCM database and grant subsequent access specified in the *dwDesiredAccess* parameter of the client request to clients using this handle after evaluating the client security context against SCM SecurityDescriptor. The server MUST return this handle by setting the *lpScHandle* parameter of the client request.

If the caller cannot be granted permission requested in the *dwDesiredAccess* parameter, the server MUST fail the call.[<40>](#Appendix_A_40" \o "Product behavior note 40)

The server MUST return ERROR\_INVALID\_NAME (123) if *lpDatabaseName* is not NULL and not ServicesActive or ServicesFailed.

The server MUST return ERROR\_DATABASE\_DOES\_NOT\_EXIST (1065) if *lpDatabaseName* is ServicesFailed.

#### ROpenServiceW (Opnum 16)

The ROpenServiceW method creates an RPC context handle to an existing [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721).

1. DWORD ROpenServiceW(
2. [in] SC\_RPC\_HANDLE hSCManager,
3. [in, string, range(0, SC\_MAX\_NAME\_LENGTH)]
4. wchar\_t\* lpServiceName,
5. [in] DWORD dwDesiredAccess,
6. [out] LPSC\_RPC\_HANDLE lpServiceHandle
7. );

**hSCManager:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database, created using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282).

**lpServiceName:** A pointer to a null-terminated **UNICODE** string that specifies the ServiceName of the service record.

The forward slash, back slash, comma, and space characters are illegal in service names.

**dwDesiredAccess:** A value that specifies the access right. This MUST be one of the values as specified in section 3.1.4.

**lpServiceHandle:** An LPSC\_RPC\_HANDLE (section 2.2.4) data type that defines the handle to the found service record.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The access specified by the *dwDesiredAccess* parameter cannot be granted to the caller. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 123  ERROR\_INVALID\_NAME | The specified service name is invalid. |
| 1060  ERROR\_SERVICE\_DOES\_NOT\_EXIST | The service record with a specified DisplayName does not exist in the SCM database. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST create an RPC context handle to the service record identified by the *lpServiceName* parameter in the SCM database identified by the *hSCManager* parameter of the client request after evaluating the SecurityDescriptor found in the service record against the caller's security context for the requested access. The server MUST increment the **HandleCount** field of the service record and return this handle by setting the *lpScHandle* parameter.

#### RQueryServiceConfigW (Opnum 17)

The RQueryServiceConfigW method returns the configuration parameters of the specified service.

1. DWORD RQueryServiceConfigW(
2. [in] SC\_RPC\_HANDLE hService,
3. [out] LPQUERY\_SERVICE\_CONFIGW lpServiceConfig,
4. [in, range(0, 1024\*8)] DWORD cbBufSize,
5. [out] LPBOUNDED\_DWORD\_8K pcbBytesNeeded
6. );

**hService:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the service record that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SERVICE\_QUERY\_CONFIG access right MUST have been granted to the caller when the RPC context handle was created.

**lpServiceConfig:** A pointer to a buffer that contains the [QUERY\_SERVICE\_CONFIGW (section 2.2.15)](#Section_97200665563142ea99176f9b41f02391) structure.

**cbBufSize:** The size, in bytes, of the *lpServiceConfig* parameter.

**pcbBytesNeeded:** An [LPBOUNDED\_DWORD\_8K (section 2.2.8)](#Section_1720ed6ed90741c68d98fb6e4877d761) data type that defines the pointer to a variable that contains the number of bytes needed to return all the configuration information if the method fails.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SERVICE\_QUERY\_CONFIG access right had not been granted to the caller when the RPC context handle was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 122  ERROR\_INSUFFICIENT\_BUFFER | The data area passed to a system call is too small. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST query the configuration information stored in the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database in the service record identified by the *hService* parameter of the client request. The server MUST return this configuration data by setting the *lpServiceConfig* parameter as specified in 2.2.15.

The server MUST set the required buffer size, in bytes, in the *pcbBytesNeeded* parameter. If the buffer pointed to by *lpServiceConfig* is insufficient to hold all the configuration data, the server MUST fail the call with ERROR\_INSUFFICIENT\_BUFFER (122).

#### RQueryServiceLockStatusW (Opnum 18)

The RQueryServiceLockStatusW method returns the lock status of the specified [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database.

1. DWORD RQueryServiceLockStatusW(
2. [in] SC\_RPC\_HANDLE hSCManager,
3. [out] LPQUERY\_SERVICE\_LOCK\_STATUSW lpLockStatus,
4. [in, range(0, 1024\*4)] DWORD cbBufSize,
5. [out] LPBOUNDED\_DWORD\_4K pcbBytesNeeded
6. );

**hSCManager:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the SCM database created using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SC\_MANAGER\_QUERY\_LOCK\_STATUS access right MUST have been granted to the caller when the RPC context handle was created.

**lpLockStatus:** A pointer to a buffer that contains [QUERY\_SERVICE\_LOCK\_STATUSW (section 2.2.17)](#Section_32df480ef28d4a68a18239692b7c9a78) structures.

**cbBufSize:** The size, in bytes, of the *lpLockStatus* buffer.

**pcbBytesNeeded:** An [LPBOUNDED\_DWORD\_4K (section 2.2.7)](#Section_d5e2cc227db64101b36ff4377e836b4c) data type that defines the pointer to a variable that receives the number of bytes needed to return all the lock status information if the method fails.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SC\_MANAGER\_QUERY\_LOCK\_STATUS access right had not been granted to the caller when the RPC context handle was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 122  ERROR\_INSUFFICIENT\_BUFFER | The data area passed to a system call is too small. |

In response to this request from the client, for a successful operation the server MUST query the lock status of the SCM database identified by the *hSCManager* parameter of the client request. The server MUST return this lock status by setting the *lpLockStatus* parameter as specified in 2.2.17.

If the buffer pointed to by *lpLockStatus* is insufficient to hold all the lock status data, the server MUST fail the call with ERROR\_INSUFFICIENT\_BUFFER (122) and set the required buffer size in the *pcbBytesNeeded* parameter. If the size is sufficient for data returned, the server also returns the required size, in bytes.

#### RStartServiceW (Opnum 19)

The RStartServiceW method starts a specified service.

1. DWORD RStartServiceW(
2. [in] SC\_RPC\_HANDLE hService,
3. [in, range(0, SC\_MAX\_ARGUMENTS)]
4. DWORD argc,
5. [in, unique, size\_is(argc)] LPSTRING\_PTRSW argv
6. );

**hService:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) that MUST have been created previously using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SERVICE\_START access right MUST have been granted to the caller when the RPC context handle to the service record was created.

**argc:** The number of argument strings in the *argv* array. If *argv* is **NULL**, this parameter MAY be 0.

**argv:** A pointer to a buffer that contains an array of pointers to null-terminated **UNICODE** strings that are passed as arguments to the service.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.[<41>](#Appendix_A_41" \o "Product behavior note 41)

| Return value/code | Description |
| --- | --- |
| 2  ERROR\_FILE\_NOT\_FOUND | The system cannot find the file specified. |
| 3  ERROR\_PATH\_NOT\_FOUND | The system cannot find the path specified. |
| 5  ERROR\_ACCESS\_DENIED | The SERVICE\_START access right had not been granted to the caller when the RPC context handle to the service record was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 1053  ERROR\_SERVICE\_REQUEST\_TIMEOUT | The process for the service was started, but it did not respond within an implementation-specific time-out.[<42>](#Appendix_A_42" \o "Product behavior note 42) |
| 1054  ERROR\_SERVICE\_NO\_THREAD | A thread could not be created for the service. |
| 1055  ERROR\_SERVICE\_DATABASE\_LOCKED | The service database is locked by the call to the **BlockServiceDatabase** method.[<43>](#Appendix_A_43" \o "Product behavior note 43) |
| 1056  ERROR\_SERVICE\_ALREADY\_RUNNING | The **ServiceStatus.dwCurrentState** in the service record is not set to SERVICE\_STOPPED. |
| 1058  ERROR\_SERVICE\_DISABLED | The service cannot be started because the Start field in the service record is set to SERVICE\_DISABLED. |
| 1068  ERROR\_SERVICE\_DEPENDENCY\_FAIL | The specified service depends on another service that has failed to start. |
| 1069  ERROR\_SERVICE\_LOGON\_FAILED | The service did not start due to a logon failure. |
| 1072  ERROR\_SERVICE\_MARKED\_FOR\_DELETE | The **RDeleteService** method has been called for the service record identified by the *hService* parameter. |
| 1075  ERROR\_SERVICE\_DEPENDENCY\_DELETED | The specified service depends on a service that does not exist or has been marked for deletion. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST start the service using the information from the service record identified by the *hService* parameter and pass the arguments specified in the *argv* parameter as part of the service launch command.

If *argv* is not NULL, the client SHOULD set the first element in *argv* to the name of the service.

The server MUST ignore *argv* for service records with **Type** equal to SERVICE\_KERNEL\_DRIVER or SERVICE\_FILE\_SYSTEM\_DRIVER.

The server MUST set the **ServiceStatus.dwCurrentState** in the service record, as specified in [SERVICE\_STATUS (section 2.2.47)](#Section_4e91ff36ab5f49eda43da308e72b0b3c), to SERVICE\_START\_PENDING.

The server MUST set the **ServiceStatus.dwControlsAccepted** in the service record, as specified in SERVICE\_STATUS, to none (zero).

The server MUST set the **ServiceStatus.dwCheckPoint** in the service record, as specified in SERVICE\_STATUS, to zero.

The server MUST set the **ServiceStatus.dwWaitHint** in the service record, as specified in SERVICE\_STATUS, to 2 seconds.

The server MUST return ERROR\_SERVICE\_NO\_THREAD if it is unable to create a new thread for the service process.

If *argv* does not contain as many non-NULL pointers as indicated by *argc*, the server MUST fail the call with ERROR\_INVALID\_PARAMETER (87).

#### RGetServiceDisplayNameW (Opnum 20)

The RGetServiceDisplayNameW method returns the display name of the specified service.

1. DWORD RGetServiceDisplayNameW(
2. [in] SC\_RPC\_HANDLE hSCManager,
3. [in, string, range(0, SC\_MAX\_NAME\_LENGTH)]
4. wchar\_t\* lpServiceName,
5. [out, string, range(1, 4\*1024+1), size\_is(\* lpcchBuffer +1)]
6. wchar\_t\* lpDisplayName,
7. [in, out] DWORD\* lpcchBuffer
8. );

**hSCManager:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database, created using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282).

**lpServiceName:** A pointer to a null-terminated **UNICODE** string that specifies the service name.

The forward slash, back slash, comma, and space characters are illegal in service names.

**lpDisplayName:** A pointer to a buffer that receives the null-terminated **UNICODE** string that contains the service display name.

**lpcchBuffer:** A **DWORD** data type that defines the pointer to a variable that specifies the size, in wchar\_ts, of the buffer. On output, this variable receives the size of the service's display name, excluding the terminating null character.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 122  ERROR\_INSUFFICIENT\_BUFFER | The display name does not fit in the buffer. |
| 123  ERROR\_INVALID\_NAME | The specified service name is invalid. |
| 1060  ERROR\_SERVICE\_DOES\_NOT\_EXIST | The [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) with the specified ServiceName does not exist in the SCM database identified by the *hSCManager* parameter. |

In response to this request from the client, for a successful operation the server MUST look up the service record with the ServiceName matching the specified *lpServiceName* in the SCM database identified by the *hSCManager* parameter. The server MUST return the DisplayName from the found service record in the *lpDisplayName* parameter and set the size in wchar\_ts of the display name excluding the terminating null character in *lpcchBuffer*.

If the *lpDisplayName* buffer is insufficient to hold the complete display name of the service, the server MUST fail the call with ERROR\_INSUFFICIENT\_BUFFER (122) and set the size in wchar\_ts of the display name excluding the terminating null character in *lpcchBuffer*. If the size is sufficient for data returned, the server also returns the required size, in bytes.

#### RGetServiceKeyNameW (Opnum 21)

The RGetServiceKeyNameW method returns the **ServiceName** of the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) with the specified **DisplayName**.

1. DWORD RGetServiceKeyNameW(
2. [in] SC\_RPC\_HANDLE hSCManager,
3. [in, string, range(0, SC\_MAX\_NAME\_LENGTH)]
4. wchar\_t\* lpDisplayName,
5. [out, string, range(1, 4\*1024+1), size\_is(\*lpcchBuffer+1)]
6. wchar\_t\* lpServiceName,
7. [in, out] DWORD\* lpcchBuffer
8. );

**hSCManager:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database created using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282).

**lpDisplayName:** A pointer to a null-terminated **UNICODE** string that specifies the service display name.

**lpServiceName:** A pointer to a buffer that receives the null-terminated **UNICODE** string that contains the service name.

The forward slash, back slash, comma, and space characters are illegal in service names.

**lpcchBuffer:** A **DWORD** data type that defines the pointer to a variable that specifies the size, in wchar\_ts, of the buffer. On output, this variable receives the size of the service name, excluding the terminating null character.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 123  ERROR\_INVALID\_NAME | The name specified in the *lpDisplayName* parameter is invalid or set to NULL. |
| 1060  ERROR\_SERVICE\_DOES\_NOT\_EXIST | The service record with the **DisplayName** matching the value specified in the *lpDisplayName* parameter does not exist in the SCM database identified by the *hSCManager* parameter. |

In response to this request from the client, for a successful operation the server MUST look up the service record with DisplayName matching the display name specified by the *lpDisplayName* parameter in the SCM database identified by *hSCManager*.

The server MUST return the ServiceName from the found service record in the *lpServiceName* parameter and set the size in wchar\_ts of the service name excluding the terminating null character in the *lpcchBuffer* parameter.

If the *lpServiceName* buffer is insufficient to hold the complete service name of the service, the server MUST fail the call with ERROR\_INSUFFICIENT\_BUFFER (122) and set the size in wchar\_ts of the service name excluding the terminating null character in the *lpcchBuffer* parameter. If the size is sufficient for data returned, the server also returns the required size, in bytes.

#### RChangeServiceConfigA (Opnum 23)

The RChangeServiceConfigA method changes a service's configuration parameters in the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database.

1. DWORD RChangeServiceConfigA(
2. [in] SC\_RPC\_HANDLE hService,
3. [in] DWORD dwServiceType,
4. [in] DWORD dwStartType,
5. [in] DWORD dwErrorControl,
6. [in, string, unique, range(0, SC\_MAX\_PATH\_LENGTH)]
7. LPSTR lpBinaryPathName,
8. [in, string, unique, range(0, SC\_MAX\_NAME\_LENGTH)]
9. LPSTR lpLoadOrderGroup,
10. [in, out, unique] LPDWORD lpdwTagId,
11. [in, unique, size\_is(dwDependSize)]
12. LPBYTE lpDependencies,
13. [in, range(0, SC\_MAX\_DEPEND\_SIZE)]
14. DWORD dwDependSize,
15. [in, string, unique, range(0, SC\_MAX\_ACCOUNT\_NAME\_LENGTH)]
16. LPSTR lpServiceStartName,
17. [in, unique, size\_is(dwPwSize)]
18. LPBYTE lpPassword,
19. [in, range(0, SC\_MAX\_PWD\_SIZE)]
20. DWORD dwPwSize,
21. [in, string, unique, range(0, SC\_MAX\_NAME\_LENGTH)]
22. LPSTR lpDisplayName
23. );

**hService:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SERVICE\_CHANGE\_CONFIG access right MUST have been granted to the caller when the [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) to the service record was created.

**dwServiceType:** A Type value for the service record (section [3.1.1](#Section_3b30d63a095b4bc184ec72370fc1db61)) that specifies the type of service. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_KERNEL\_DRIVER  0x00000001 | A driver service. These are services that manage devices on the system. |
| SERVICE\_FILE\_SYSTEM\_DRIVER  0x00000002 | A file system driver service. These are services that manage file systems on the system. |
| SERVICE\_WIN32\_OWN\_PROCESS  0x00000010 | Service that runs in its own process. |
| SERVICE\_WIN32\_SHARE\_PROCESS  0x00000020 | Service that shares a process with other services. |
| SERVICE\_NO\_CHANGE  0xFFFFFFFF | Service type does not change. |

The following flag can also be combined with the value passed in *dwServiceType*.

| Value | Meaning |
| --- | --- |
| SERVICE\_INTERACTIVE\_PROCESS  0x00000100 | The service can interact with the desktop. |

**dwStartType:** A Start value for the service record (section 3.1.1) that specifies when to start the service. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_BOOT\_START  0x00000000 | Starts the driver service when the system boots up. This value is valid only for driver services. |
| SERVICE\_SYSTEM\_START  0x00000001 | Starts the driver service when the system boots up. This value is valid only for driver services. The services marked SERVICE\_SYSTEM\_START are started after all SERVICE\_BOOT\_START services have been started. |
| SERVICE\_AUTO\_START  0x00000002 | Starts the service automatically during system startup. |
| SERVICE\_DEMAND\_START  0x00000003 | Starts the service when a client requests the SCM to start the service. |
| SERVICE\_DISABLED  0x00000004 | Service cannot be started. |
| SERVICE\_NO\_CHANGE  0xFFFFFFFF | Service start type does not change. |

**dwErrorControl:** An ErrorControl value for the service record (section 3.1.1) that specifies the severity of the error if the service fails to start and determines the action that the SCM takes. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_ERROR\_IGNORE  0x00000000 | The SCM ignores the error and continues the startup operation. |
| SERVICE\_ERROR\_NORMAL  0x00000001 | The SCM logs the error, but continues the startup operation. |
| SERVICE\_ERROR\_SEVERE  0x00000002 | The SCM 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  0x00000003 | The SCM SHOULD log 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  0xFFFFFFFF | Service error control type does not change. |

**lpBinaryPathName:** An ImagePath value for the service record (section 3.1.1) as a pointer to a null-terminated ANSI string that contains the fully qualified path to the service binary file. The path MAY include arguments. If the path contains a space, it MUST be quoted so that it is correctly interpreted. For example, "d:\\my share\\myservice.exe" is specified as "\"d:\\my share\\myservice.exe\"".

**lpLoadOrderGroup:** A Group value for the service record (section 3.1.1) as a pointer to a null-terminated ANSI string that names the load ordering group of which this service is a member.

Specify NULL or an empty string if the service does not belong to a [**load-ordering group**](#gt_88c26bf0-5b6f-4423-82b3-1027ea5df0e3).

**lpdwTagId:** A Tag value for the service record (section 3.1.1) as a pointer to a variable that receives a tag value. The value is unique to the group specified in the *lpLoadOrderGroup* parameter.

**lpDependencies:** DependOnSize and DependOnGroup values for the service record (section 3.1.1) as a pointer to an array of null-separated names of services or load ordering groups that MUST start before this service. The array is doubly null-terminated. Load ordering group names are prefixed with a "+" character (to distinguish them from service names). If the pointer is NULL or if it points to an empty string, the service has no dependencies. Cyclic dependency between services is not allowed. The character set is [**ANSI**](#gt_100cd8a6-5cb1-4895-9de6-e4a3c224a583). 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.

**dwDependSize:** The size, in bytes, of the string specified by the *lpDependencies* parameter.

**lpServiceStartName:** An ObjectName value for the service record (section 3.1.1) as a pointer to a null-terminated ANSI string that specifies the name of the account under which the service runs.

**lpPassword:** A Password value for the service record (section 3.1.1) as a pointer to a null-terminated ANSI string that contains the password of the account whose name was specified by the *lpServiceStartName* parameter.

**dwPwSize:** The size, in bytes, of the password specified by the *lpPassword* parameter.

**lpDisplayName:** A DisplayName value for the service record (section 3.1.1) as a pointer to a null-terminated ANSI string that contains the display name that applications can use to identify the service for its users.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SERVICE\_CHANGE\_CONFIG access right had not been granted to the caller when the RPC context handle to the service record was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle specified is invalid. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 1057  ERROR\_INVALID\_SERVICE\_ACCOUNT | The user account name specified in the *lpServiceStartName* parameter does not exist. |
| 1059  ERROR\_CIRCULAR\_DEPENDENCY | A circular service dependency was specified. |
| 1078  ERROR\_DUPLICATE\_SERVICE\_NAME | The *lpDisplayName* matches either the ServiceName or the DisplayName of another service record in the service control manager database. |
| 1072  ERROR\_SERVICE\_MARKED\_FOR\_DELETE | The RDeleteService has been called for the service record identified by the *hService* parameter. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST update, using the values from the appropriate parameters of the client request, the service record identified by the *hService* parameter in the SCM database:

* If the client passes NULL for *lpBinaryPathName*, the server MUST keep the existing ImagePath value.
* If the client passes NULL for *lpLoadOrderGroup*, the server MUST keep the existing ServiceGroup value.
* If the client passes NULL for *lpdwTagId*, the server MUST keep the existing Tag value.
* If the client passes NULL for *lpDependencies*, the server MUST keep the existing DependOnService and DependOnGroup values.
* If the client passes NULL for *lpServiceStartName*, the server MUST keep the existing ObjectName value.
* If the client passes NULL for *lpPassword*, the server MUST keep the existing Password value.
* If the client passes NULL for *lpDisplayName*, the server MUST keep the existing DisplayName value.

If the original service type is SERVICE\_WIN32\_OWN\_PROCESS or SERVICE\_WIN32\_SHARE\_PROCESS, the server MUST fail the call if *dwServiceType* is set to SERVICE\_FILE\_SYSTEM\_DRIVER or SERVICE\_KERNEL\_DRIVER.[<44>](#Appendix_A_44" \o "Product behavior note 44)

If *dwServiceType* is set to SERVICE\_WIN32\_OWN\_PROCESS or SERVICE\_WIN32\_SHARE\_PROCESS combined with the SERVICE\_INTERACTIVE\_PROCESS bit and the **ObjectName** field of the service record is not equal to LocalSystem, the server MUST fail the request with ERROR\_INVALID\_PARAMETER.

If the service has a PreferredNode setting and the client requested a change in service type other than SERVICE\_WIN32\_OWN\_PROCESS, the server MUST fail the call with ERROR\_INVALID\_PARAMETER (87).

If the service is a member of a load-order group and has a start type of delayed autostart (see section [2.2.33](#Section_805b8296863d4d1e8ae8f639adf8c6cb)), then the server MUST fail the call with ERROR\_INVALID\_PARAMETER (87).

If *lpdwTagId* has a valid value and *lpLoadOrderGroup* is either NULL or an empty string, then the server MUST return ERROR\_INVALID\_PARAMETER.

The server MUST use the process described in [Conversion Between ANSI and Unicode String Formats (section 3.1.7)](#Section_f9ddeebeb05e423c944084fd3d14bee9) to convert a string to the appropriate format.

For service record changes to apply to the running service, the service MUST be stopped and started back up, except in the case of *lpDisplayName*. Changes to *lpDisplayName* take effect immediately.

If *lpBinaryPathName* contains arguments, the server MUST pass these arguments to the service entry point.

#### RCreateServiceA (Opnum 24)

The RCreateServiceA method creates the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) in the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database.

1. DWORD RCreateServiceA(
2. [in] SC\_RPC\_HANDLE hSCManager,
3. [in, string, range(0, SC\_MAX\_NAME\_LENGTH)]
4. LPSTR lpServiceName,
5. [in, string, unique, range(0, SC\_MAX\_NAME\_LENGTH)]
6. LPSTR lpDisplayName,
7. [in] DWORD dwDesiredAccess,
8. [in] DWORD dwServiceType,
9. [in] DWORD dwStartType,
10. [in] DWORD dwErrorControl,
11. [in, string, range(0, SC\_MAX\_PATH\_LENGTH)]
12. LPSTR lpBinaryPathName,
13. [in, string, unique, range(0, SC\_MAX\_NAME\_LENGTH)]
14. LPSTR lpLoadOrderGroup,
15. [in, out, unique] LPDWORD lpdwTagId,
16. [in, unique, size\_is(dwDependSize)]
17. LPBYTE lpDependencies,
18. [in, range(0, SC\_MAX\_DEPEND\_SIZE)]
19. DWORD dwDependSize,
20. [in, string, unique, range(0, SC\_MAX\_ACCOUNT\_NAME\_LENGTH)]
21. LPSTR lpServiceStartName,
22. [in, unique, size\_is(dwPwSize)]
23. LPBYTE lpPassword,
24. [in, range(0, SC\_MAX\_PWD\_SIZE)]
25. DWORD dwPwSize,
26. [out] LPSC\_RPC\_HANDLE lpServiceHandle
27. );

**hSCManager:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the SCM database created using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SC\_MANAGER\_CREATE\_SERVICE access right MUST have been granted to the caller when the RPC context handle was created.

**lpServiceName:** A pointer to a null-terminated ANSI string that specifies the name of the service to install. This MUST not be null.

The forward slash, back slash, comma, and space characters are illegal in service names.

**lpDisplayName:** A pointer to a null-terminated ANSI string that contains the display name by which user interface programs identify the service.

**dwDesiredAccess:** A value that specifies the access to the service. This MUST be one of the values specified in section 3.1.4.

The following generic access types also can be specified.

**dwServiceType:** A value that specifies the type of service. This MUST be one or a combination of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_KERNEL\_DRIVER  0x00000001 | A driver service. These are services that manage devices on the system. |
| SERVICE\_FILE\_SYSTEM\_DRIVER  0x00000002 | A file system driver service. These are services that manage file systems on the system. |
| SERVICE\_WIN32\_OWN\_PROCESS  0x00000010 | Service that runs in its own process. |
| SERVICE\_WIN32\_SHARE\_PROCESS  0x00000020 | Service that shares a process with other services. |
| SERVICE\_INTERACTIVE\_PROCESS  0x00000100 | The service can interact with the desktop. |

**dwStartType:** A value that specifies when to start the service. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_BOOT\_START  0x00000000 | Starts the driver service when the system boots up. This value is valid only for driver services. |
| SERVICE\_SYSTEM\_START  0x00000001 | Starts the driver service when the system boots up. This value is valid only for driver services. The services marked SERVICE\_SYSTEM\_START are started after all SERVICE\_BOOT\_START services have been started. |
| SERVICE\_AUTO\_START  0x00000002 | Starts the service automatically during system startup. |
| SERVICE\_DEMAND\_START  0x00000003 | The SCM starts the service when a process calls the StartService function. For more information, see [[MSDN-STARTSERVICE]](https://go.microsoft.com/fwlink/?LinkId=90137). |
| SERVICE\_DISABLED  0x00000004 | Service cannot be started. |

**dwErrorControl:** A value that specifies the severity of the error if the service fails to start and determines the action that the SCM takes. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_ERROR\_IGNORE  0x00000000 | The SCM ignores the error and continues the startup operation. |
| SERVICE\_ERROR\_NORMAL  0x00000001 | The SCM logs the error, but continues the startup operation. |
| SERVICE\_ERROR\_SEVERE  0x00000002 | The SCM 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  0x00000003 | The SCM SHOULD log 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. |

**lpBinaryPathName:** A pointer to a null-terminated ANSI string that contains the fully qualified path to the service binary file. The path MAY include arguments. If the path contains a space, it MUST be quoted so that it is correctly interpreted. For example, "d:\\my share\\myservice.exe" is specified as "\"d:\\my share\\myservice.exe\"".

**lpLoadOrderGroup:** A pointer to a null-terminated ANSI string that names the load-ordering group of which this service is a member.

Specify NULL or an empty string if the service does not belong to a [**load-ordering group**](#gt_88c26bf0-5b6f-4423-82b3-1027ea5df0e3).

**lpdwTagId:** A pointer to a variable that receives a tag value. The value is unique to the group specified in the *lpLoadOrderGroup* parameter.

**lpDependencies:** A pointer to an array of null-separated names of services or load ordering groups that MUST start before this service. The array is doubly null-terminated. Load ordering group names are prefixed with a "+" character (to distinguish them from service names). If the pointer is NULL or if it points to an empty string, the service has no dependencies. Cyclic dependency between services is not allowed. The character set is [**ANSI**](#gt_100cd8a6-5cb1-4895-9de6-e4a3c224a583). 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.

**dwDependSize:** The size, in bytes, of the string specified by the *lpDependencies* parameter.

**lpServiceStartName:** A pointer to a null-terminated ANSI string that specifies the name of the account under which the service runs.

**lpPassword:** A pointer to a null-terminated ANSI string that contains the password of the account whose name was specified by the *lpServiceStartName* parameter.

**dwPwSize:** The size, in bytes, of the password specified by the *lpPassword* parameter.

**lpServiceHandle:** An LPSC\_RPC\_HANDLE (section 2.2.4) data type that defines the handle to the newly created service record.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SC\_MANAGER\_CREATE\_SERVICE access right had not been granted to the caller when the RPC context handle was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle specified is invalid. |
| 13  ERROR\_INVALID\_DATA | The data is invalid. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 123  ERROR\_INVALID\_NAME | The specified service name is invalid. |
| 1057  ERROR\_INVALID\_SERVICE\_ACCOUNT | The user account name specified in the *lpServiceStartName* parameter does not exist. |
| 1059  ERROR\_CIRCULAR\_DEPENDENCY | A circular service dependency was specified. |
| 1072  ERROR\_SERVICE\_MARKED\_FOR\_DELETE | The service record with a specified name already exists, and RDeleteService has been called for it. |
| 1073  ERROR\_SERVICE\_EXISTS | The service record with the ServiceName matching the specified *lpServiceName* already exists. |
| 1078  ERROR\_DUPLICATE\_SERVICE\_NAME | The service record with the same DisplayName or the same ServiceName as the passed-in *lpDisplayName* already exists in the service control manager database. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST use the service name specified in the *lpServiceName* parameter to create a new service record in the SCM database and use the values from the appropriate parameters of the client request to update the attributes of this newly created service record.

The server MUST treat the lpPassword as a clear-text password if the client is using RPC over TCP, ncacn\_ip\_tcp (as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)). See section [2.1.2](#Section_1901c4c33d6e426989cc42e21f0e9892) Client.

The server MUST treat the lpPassword as encrypted and decrypt it, if the client is using a RPC over NP, ncacn\_np (as specified in [MS-RPCE]). The server MUST first retrieve a [**session key**](#gt_4f67a585-fb00-4166-93e8-cf4abca8226d) as specified in [[MS-CIFS]](%5bMS-CIFS%5d.pdf#Section_d416ff7cc536406ea9514f04b2fd1d2b) (section 3.5.4.4). An [**RPC server**](#gt_ae65dac0-cd24-4e83-a946-6d1097b71553) application requests the session key of a client and then uses the routine as specified in [[MS-LSAD]](%5bMS-LSAD%5d.pdf#Section_1b5471ef4c334a91b079dfcbb82f05cc) (section 5.1.2) to decrypt the password.

If the service is created successfully, the server MUST return a handle to the service in the *lpServiceHandle* parameter with the access rights associated with this handle as specified in the *dwDesiredAccess* parameter of the client request.

The server MUST use the process described in [Conversion Between ANSI and Unicode String Formats (section 3.1.7)](#Section_f9ddeebeb05e423c944084fd3d14bee9) to convert a string to the appropriate format.

If the service is created successfully, the server MUST increment the **HandleCount** field of the service record.

The only valid combinations of values for *dwServiceType* are SERVICE\_INTERACTIVE\_PROCESS and SERVICE\_WIN32\_OWN\_PROCESS or SERVICE\_INTERACTIVE\_PROCESS and SERVICE\_WIN32\_SHARE\_PROCESS. If the value of *dwServiceType* has more than one bit set and the combination of bits is not equal to SERVICE\_INTERACTIVE\_PROCESS and SERVICE\_WIN32\_OWN\_PROCESS or SERVICE\_INTERACTIVE\_PROCESS and SERVICE\_WIN32\_SHARE\_PROCESS, the server MUST fail the method and return the error ERROR\_INVALID\_PARAMETER.

If *lpBinaryPathName* contains arguments, the server MUST pass these arguments to the service entry point.

If *lpdwTagId* has a valid value and *lpLoadOrderGroup* is either NULL or an empty string, then the server MUST return ERROR\_INVALID\_PARAMETER.

#### REnumDependentServicesA (Opnum 25)

The REnumDependentServicesA method returns the ServiceName, DisplayName, and ServiceStatus of each [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) that depends on the specified service.

1. DWORD REnumDependentServicesA(
2. [in] SC\_RPC\_HANDLE hService,
3. [in] DWORD dwServiceState,
4. [out, size\_is(cbBufSize)] LPBYTE lpServices,
5. [in, range(0, 1024\*256)] DWORD cbBufSize,
6. [out] LPBOUNDED\_DWORD\_256K pcbBytesNeeded,
7. [out] LPBOUNDED\_DWORD\_256K lpServicesReturned
8. );

**hService:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the service record that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SERVICE\_ENUMERATE\_DEPENDENT access right MUST have been granted to the caller when the [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) to the service record was created.

**dwServiceState:** A value that specifies the service records to enumerate based on the value of their ServiceStatus.dwCurrentState. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_ACTIVE  0x00000001 | Enumerates service records that have ServiceStatus.dwCurrentState equal to one of the following: SERVICE\_START\_PENDING, SERVICE\_STOP\_PENDING, SERVICE\_RUNNING, SERVICE\_CONTINUE\_PENDING, SERVICE\_PAUSE\_PENDING, and SERVICE\_PAUSED. |
| SERVICE\_INACTIVE  0x00000002 | Enumerates service records that have ServiceStatus.dwCurrentState equal to SERVICE\_STOPPED. |
| SERVICE\_STATE\_ALL  0x00000003 | Enumerates services that have ServiceStatus.dwCurrentState equal to one of the following: SERVICE\_START\_PENDING, SERVICE\_STOP\_PENDING, SERVICE\_RUNNING, SERVICE\_CONTINUE\_PENDING, SERVICE\_PAUSE\_PENDING, SERVICE\_PAUSED, and SERVICE\_STOPPED. |

**lpServices:** A pointer to an array of [ENUM\_SERVICE\_STATUSA (section 2.2.10)](#Section_867593839e734b46aeb973d21c2d1f6c) structures that contain the name and service status information for each dependent service record in the database.

**cbBufSize:** The size, in bytes, of the array pointed to by *lpServices*.

**pcbBytesNeeded:** An [LPBOUNDED\_DWORD\_256K (section 2.2.9)](#Section_6c441817d7364a94a4440c1810a6c473) pointer to a variable that contains the number of bytes needed to store the array of service entries.

**lpServicesReturned:** An LPBOUNDED\_DWORD\_256K (section 2.2.9) pointer to a variable that contains the number of service entries returned.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SERVICE\_ENUMERATE\_DEPENDENT access right had not been granted to the caller when the RPC context handle to the service record was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 234  ERROR\_MORE\_DATA | More data is available. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation, the server MUST determine the list of service records that depend on the service record identified by the *hService* parameter of the client request. The server MUST return this list by setting the ServiceName, DisplayName, and ServiceStatus of each service record in this list in the array of ENUM\_SERVICE\_STATUSA (section 2.2.10) structures pointed to by the *lpServices* parameter and MUST set the number of services returned in the *lpServicesReturned* parameter.

If the size of the *lpServices* array is insufficient for the list of services returned, the server MUST fail the call with ERROR\_MORE\_DATA (234) and return the size in bytes required in the *pcbBytesNeeded* parameter. If the size is sufficient for data returned, the server also returns the required size, in bytes.

If the size of the *lpServices* array is sufficient for the list of services returned, the enumerated data MAY be in the buffer in a non-contiguous manner, and portions of the *lpServices* array MAY be empty (filled with 0x00).

The server MUST use the process described in [Conversion Between ANSI and Unicode String Formats (section 3.1.7)](#Section_f9ddeebeb05e423c944084fd3d14bee9) to convert a string to the appropriate format.

The server MUST return the services in reverse sequence of the start order of the services.

The server MUST return ERROR\_INVALID\_PARAMETER (87) if a bitmask specified in *dwServiceState* contains undefined values.

#### REnumServicesStatusA (Opnum 26)

The REnumServicesStatusA method enumerates [**service records**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) in the specified [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database.

1. DWORD REnumServicesStatusA(
2. [in] SC\_RPC\_HANDLE hSCManager,
3. [in] DWORD dwServiceType,
4. [in] DWORD dwServiceState,
5. [out, size\_is(cbBufSize)] LPBYTE lpBuffer,
6. [in, range(0, 1024\*256)] DWORD cbBufSize,
7. [out] LPBOUNDED\_DWORD\_256K pcbBytesNeeded,
8. [out] LPBOUNDED\_DWORD\_256K lpServicesReturned,
9. [in, out, unique] LPBOUNDED\_DWORD\_256K lpResumeIndex
10. );

**hSCManager:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the SCM database that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SC\_MANAGER\_ENUMERATE\_SERVICE access right MUST have been granted to the caller when the [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) to the service record was created.

**dwServiceType:** A value that specifies the service records to enumerate based on the Type value. This MUST be one or a combination of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_KERNEL\_DRIVER  0x00000001 | A driver service. These are services that manage devices on the system. |
| SERVICE\_FILE\_SYSTEM\_DRIVER  0x00000002 | A file system driver service. These are services that manage file systems on the system. |
| SERVICE\_WIN32\_OWN\_PROCESS  0x00000010 | Service that runs in its own process. |
| SERVICE\_WIN32\_SHARE\_PROCESS  0x00000020 | Service that shares a process with other services. |

**dwServiceState:** A value that specifies the service records to enumerate based on their ServiceStatus.dwCurrentState. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_ACTIVE  0x00000001 | Enumerates service records that have ServiceStatus.dwCurrentState equal to one of the following: SERVICE\_START\_PENDING, SERVICE\_STOP\_PENDING, SERVICE\_RUNNING, SERVICE\_CONTINUE\_PENDING, SERVICE\_PAUSE\_PENDING, and SERVICE\_PAUSED. |
| SERVICE\_INACTIVE  0x00000002 | Enumerates services that have ServiceStatus.dwCurrentState equal to SERVICE\_STOPPED. |
| SERVICE\_STATE\_ALL  0x00000003 | Enumerates services that have ServiceStatus.dwCurrentState equal to one of the following: SERVICE\_START\_PENDING, SERVICE\_STOP\_PENDING, SERVICE\_RUNNING, SERVICE\_CONTINUE\_PENDING, SERVICE\_PAUSE\_PENDING, SERVICE\_PAUSED, and SERVICE\_STOPPED. |

**lpBuffer:** A pointer to an array of [ENUM\_SERVICE\_STATUSA (section 2.2.10)](#Section_867593839e734b46aeb973d21c2d1f6c) structures that contain the name and service status information for each dependent service in the database.

**cbBufSize:** The size, in bytes, of the array pointed to by *lpBuffer*.

**pcbBytesNeeded:** An [LPBOUNDED\_DWORD\_256K (section 2.2.9)](#Section_6c441817d7364a94a4440c1810a6c473) pointer to a variable that contains the number of bytes needed to store the array of service entries.

**lpServicesReturned:** An LPBOUNDED\_DWORD\_256K (section 2.2.9) pointer to a variable that contains the number of service entries returned.

**lpResumeIndex:** An LPBOUNDED\_DWORD\_256K (section 2.2.9) pointer to a variable that specifies the current position in the status enumeration. The server MUST assign a unique number to each service for the boot session, in increasing order, and increment that number by one for each service addition. The value of the *lpResumeIndex* parameter is one of these numbers, which the server can use to determine the resumption point for the enumeration.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SC\_MANAGER\_ENUMERATE\_SERVICE access right had not been granted to the caller when the RPC context handle to the service record was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 234  ERROR\_MORE\_DATA | More data is available. |

In response to this request from the client, for a successful operation the server MUST determine the list of service records in the SCM database identified by the *hSCManager* parameter with the current value of ServiceStatus.dwCurrentState equal to the state specified by the *dwServiceState* parameter and Type equal to the *dwServiceType* of the client request. The server MUST return this list by setting the ServiceName, DisplayName, and ServiceStatus of each service in this list in the array of ENUM\_SERVICE\_STATUSA (section 2.2.10) structures pointed to by the *lpServices* parameter and MUST set the number of services returned in the *lpServicesReturned* parameter.

If the *lpResumeIndex* value is not zero, the server MUST use that as the offset to the service list and return only services starting at this offset. If the *lpResumeIndex* value is zero, the server MUST return all services. The server MUST set this parameter to zero if the operation succeeds. If the *lpResumeIndex* value is set by the client to any nonzero number not returned by the server, the behavior is not defined.

If the size of the *lpServices* array is insufficient for the list of services returned, the server MUST fail the call with ERROR\_MORE\_DATA (234) and return the size in bytes required in the *pcbBytesNeeded* parameter. If the size is sufficient for data returned, the server also returns the required size, in bytes.

If the size of the *lpServices* array is sufficient for the list of services returned, the enumerated data MAY be in the buffer in a non-contiguous manner, and portions of the *lpServices* array MAY be empty (filled with 0x00).

The server MUST use the process described in [Conversion Between ANSI and Unicode String Formats (section 3.1.7)](#Section_f9ddeebeb05e423c944084fd3d14bee9) to convert a string to the appropriate format.

The server MUST return ERROR\_INVALID\_PARAMETER (87) if a bitmask specified in *dwServiceState* is zero or contains undefined values.

The server MUST return ERROR\_INVALID\_PARAMETER (87) if a bitmask specified in *dwServiceType* is zero or contains undefined values.

#### ROpenSCManagerA (Opnum 27)

The ROpenSCManagerA method opens a connection to the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) from the client and then opens the specified SCM database.

1. DWORD ROpenSCManagerA(
2. [in, string, unique, range(0, SC\_MAX\_COMPUTER\_NAME\_LENGTH)]
3. SVCCTL\_HANDLEA lpMachineName,
4. [in, string, unique, range(0, SC\_MAX\_NAME\_LENGTH)]
5. LPSTR lpDatabaseName,
6. [in] DWORD dwDesiredAccess,
7. [out] LPSC\_RPC\_HANDLE lpScHandle
8. );

**lpMachineName:** An [SVCCTL\_HANDLEA (section 2.2.2)](#Section_7e6a73a5f7954ceea230cf4056eef4d6) data type that defines the pointer to a null-terminated ANSI string that specifies the server's machine name.

**lpDatabaseName:** A pointer to a null-terminated ANSI string that specifies the name of the SCM database to open. The parameter MUST be set to NULL, "ServicesActive", or "ServicesFailed".

**dwDesiredAccess:** A value that specifies the access to the database. This MUST be one of the values specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282).

The client MUST also have the SC\_MANAGER\_CONNECT access right.

**lpScHandle:** An [LPSC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the newly opened SCM connection.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SC\_MANAGER\_CONNECT access right or the desired access is not granted to the caller in the SCM SecurityDescriptor. |
| 123  ERROR\_INVALID\_NAME | The specified service name is invalid. |
| 1065  ERROR\_DATABASE\_DOES\_NOT\_EXIST | The database specified does not exist. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST create an RPC context handle to the SCM database and grant subsequent access specified in the *dwDesiredAccess* parameter of the client request after evaluating the client security context against the SCM SecurityDescriptor. The server MUST return this handle by setting the *lpScHandle* parameter of the client request.

If the caller cannot be granted permission requested in the *dwDesiredAccess* parameter, the server MUST fail the call.[<45>](#Appendix_A_45" \o "Product behavior note 45)

The server MUST use the process described in [Conversion Between ANSI and Unicode String Formats (section 3.1.7)](#Section_f9ddeebeb05e423c944084fd3d14bee9) to convert a string to the appropriate format.

The server MUST return ERROR\_INVALID\_NAME (123) if *lpDatabaseName* is not NULL and is not ServicesActive or ServicesFailed.

The server MUST return ERROR\_DATABASE\_DOES\_NOT\_EXIST (1065) if *lpDatabaseName* is ServicesFailed.

#### ROpenServiceA (Opnum 28)

The ROpenServiceA method creates an RPC context handle to an existing [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721).

1. DWORD ROpenServiceA(
2. [in] SC\_RPC\_HANDLE hSCManager,
3. [in, string, range(0, SC\_MAX\_NAME\_LENGTH)]
4. LPSTR lpServiceName,
5. [in] DWORD dwDesiredAccess,
6. [out] LPSC\_RPC\_HANDLE lpServiceHandle
7. );

**hSCManager:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database, created using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282).

**lpServiceName:** A pointer to a null-terminated ANSI string that specifies the ServiceName of the service record to open.

The forward slash, back slash, comma, and space characters are illegal in service names.

**dwDesiredAccess:** A value that specifies the access right. This MUST be one of the values specified in section 3.1.4.

**lpServiceHandle:** An LPSC\_RPC\_HANDLE (section 2.2.4) data type that defines the handle to the found service record.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The access specified by the *dwDesiredAccess* parameter cannot be granted to the caller. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 123  ERROR\_INVALID\_NAME | The specified service name is invalid. |
| 1060  ERROR\_SERVICE\_DOES\_NOT\_EXIST | The service record with a specified DisplayName does not exist in the SCM database. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST create an RPC context handle to the service record identified by the *lpServiceName* parameter in the SCM database specified by the *hSCManager* parameter of the client request after evaluating the SecurityDescriptor found in the service record against the caller's security context for the requested access. The server MUST increment the **HandleCount** field of the service record and return this handle by setting the *lpScHandle* parameter.

The server MUST use the process described in [Conversion Between ANSI and Unicode String Formats (section 3.1.7)](#Section_f9ddeebeb05e423c944084fd3d14bee9) to convert a string to the appropriate format.

#### RQueryServiceConfigA (Opnum 29)

The RQueryServiceConfigA method returns the configuration parameters of the specified service.

1. DWORD RQueryServiceConfigA(
2. [in] SC\_RPC\_HANDLE hService,
3. [out] LPQUERY\_SERVICE\_CONFIGA lpServiceConfig,
4. [in, range(0, 1024\*8)] DWORD cbBufSize,
5. [out] LPBOUNDED\_DWORD\_8K pcbBytesNeeded
6. );

**hService:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SERVICE\_QUERY\_CONFIG access right MUST have been granted to the caller when the [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) was created.

**lpServiceConfig:** A pointer to a buffer that contains the [QUERY\_SERVICE\_CONFIGA](#Section_daee07f590754534a6747db7fc689b36) structure.

**cbBufSize:** The size, in bytes, of the *lpServiceConfig* parameter.

**pcbBytesNeeded:** An [LPBOUNDED\_DWORD\_8K (section 2.2.8)](#Section_1720ed6ed90741c68d98fb6e4877d761) data type that defines the pointer to a variable that contains the number of bytes needed to return all the configuration information if the function fails.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SERVICE\_QUERY\_CONFIG access right had not been granted to the caller when the RPC context handle was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 122  ERROR\_INSUFFICIENT\_BUFFER | The data area passed to a system call is too small. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST query the configuration information stored in the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database in the service record identified by the *hService* parameter of the client request. The server MUST return this configuration data by setting the *lpServiceConfig* parameter as specified in 2.2.14.

The server MUST set the required buffer size, in bytes, in the *pcbBytesNeeded* parameter. If the buffer pointed to by *lpServiceConfig* is insufficient to hold all the configuration data, the server MUST fail the call with ERROR\_INSUFFICIENT\_BUFFER (122).

The server MUST use the process described in [Conversion Between ANSI and Unicode String Formats (section 3.1.7)](#Section_f9ddeebeb05e423c944084fd3d14bee9) to convert a string to the appropriate format.

#### RQueryServiceLockStatusA (Opnum 30)

The RQueryServiceLockStatusA method returns the lock status of the specified [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database.

1. DWORD RQueryServiceLockStatusA(
2. [in] SC\_RPC\_HANDLE hSCManager,
3. [out] LPQUERY\_SERVICE\_LOCK\_STATUSA lpLockStatus,
4. [in, range(0, 1024\*4)] DWORD cbBufSize,
5. [out] LPBOUNDED\_DWORD\_4K pcbBytesNeeded
6. );

**hSCManager:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the SCM database created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SC\_MANAGER\_QUERY\_LOCK\_STATUS access right MUST have been granted to the caller when the [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) was created.

**lpLockStatus:** A pointer to a buffer that contains the [QUERY\_SERVICE\_LOCK\_STATUSA (section 2.2.16)](#Section_b61893cbaef2427eabf66f3b072f629d) structures.

**cbBufSize:** The size, in bytes, of the *lpLockStatus* buffer.

**pcbBytesNeeded:** An [LPBOUNDED\_DWORD\_4K (section 2.2.7)](#Section_d5e2cc227db64101b36ff4377e836b4c) data type that defines the pointer to a variable that receives the number of bytes needed to return all the lock status.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SC\_MANAGER\_QUERY\_LOCK\_STATUS access right had not been granted to the caller when the RPC context handle was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 122  ERROR\_INSUFFICIENT\_BUFFER | The data area passed to a system call is too small. |

In response to this request from the client, for a successful operation the server MUST query the lock status of the SCM database identified by the *hSCManager* parameter of the client request. The server MUST return this lock status by setting the *lpLockStatus* parameter as specified in section 2.2.16.

If the buffer pointed to by *lpLockStatus* is insufficient to hold all the lock status data, the server MUST fail the call with ERROR\_INSUFFICIENT\_BUFFER (122) and set the required buffer size in the *pcbBytesNeeded* parameter. If the size is sufficient for data returned, the server also returns the required size, in bytes.

The server MUST use the process described in [Conversion Between ANSI and Unicode String Formats (section 3.1.7)](#Section_f9ddeebeb05e423c944084fd3d14bee9) to convert a string to the appropriate format.

#### RStartServiceA (Opnum 31)

The RStartServiceA method starts a specified service.

1. DWORD RStartServiceA(
2. [in] SC\_RPC\_HANDLE hService,
3. [in, range(0, SC\_MAX\_ARGUMENTS)]
4. DWORD argc,
5. [in, unique, size\_is(argc)] LPSTRING\_PTRSA argv
6. );

**hService:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) that defines the handle to the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SERVICE\_START access right MUST have been granted to the caller when the [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) was created.

**argc:** The number of argument strings in the *argv* array. If *argv* is NULL, this parameter MAY be zero.

**argv:** A pointer to a buffer that contains an array of pointers to null-terminated ANSI strings that are passed as arguments to the service.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.[<46>](#Appendix_A_46" \o "Product behavior note 46)

| Return value/code | Description |
| --- | --- |
| 2  ERROR\_FILE\_NOT\_FOUND | The system cannot find the file specified. |
| 3  ERROR\_PATH\_NOT\_FOUND | The system cannot find the path specified. |
| 5  ERROR\_ACCESS\_DENIED | The SERVICE\_START access right had not been granted to the caller when the RPC context handle to the service was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 1053  ERROR\_SERVICE\_REQUEST\_TIMEOUT | The process for the service was started, but it did not respond within an implementation-specific time-out.[<47>](#Appendix_A_47" \o "Product behavior note 47) |
| 1054  ERROR\_SERVICE\_NO\_THREAD | A thread could not be created for the service. |
| 1055  ERROR\_SERVICE\_DATABASE\_LOCKED | The service database is locked by the call to the [RLockServiceDatabase (section 3.1.4.4)](#Section_ff71f732e91d41898fb9a410674c63ad) method.[<48>](#Appendix_A_48" \o "Product behavior note 48) |
| 1056  ERROR\_SERVICE\_ALREADY\_RUNNING | The **ServiceStatus.dwCurrentState** in the service record is not set to SERVICE\_STOPPED. |
| 1058  ERROR\_SERVICE\_DISABLED | The service cannot be started because the Start field in the service record is set to SERVICE\_DISABLED. |
| 1068  ERROR\_SERVICE\_DEPENDENCY\_FAIL | The specified service depends on another service that has failed to start. |
| 1069  ERROR\_SERVICE\_LOGON\_FAILED | The service did not start due to a logon failure. |
| 1072  ERROR\_SERVICE\_MARKED\_FOR\_DELETE | The **RDeleteService** method has been called for the service record identified by the *hService* parameter. |
| 1075  ERROR\_SERVICE\_DEPENDENCY\_DELETED | The specified service depends on a service that does not exist or has been marked for deletion. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST start the service using the information from the service record identified by the hService parameter and pass the arguments specified in the argv parameter as part of the service launch command.

If *argv* is not NULL, the client SHOULD set the first element in *argv* to the name of the service.

The server MUST ignore *argv* for service records with **Type** equal to SERVICE\_KERNEL\_DRIVER or SERVICE\_FILE\_SYSTEM\_DRIVER.

The server MUST set the **ServiceStatus.dwCurrentState** in the service record, as specified in [SERVICE\_STATUS (section 2.2.47)](#Section_4e91ff36ab5f49eda43da308e72b0b3c), to SERVICE\_START\_PENDING.

The server MUST set the **ServiceStatus.dwControlsAccepted** in the service record, as specified in SERVICE\_STATUS, to none (zero).

The server MUST set the **ServiceStatus.dwCheckPoint** in the service record, as specified in SERVICE\_STATUS, to zero.

The server MUST set the **ServiceStatus.dwWaitHint** in the service record, as specified in SERVICE\_STATUS, to 2 seconds.

The server MUST return ERROR\_SERVICE\_NO\_THREAD if it is unable to create a new thread for the service process.

If *argv* does not contain as many non-NULL pointers as indicated by *argc*, the server MUST fail the call with ERROR\_INVALID\_PARAMETER (87).

#### RGetServiceDisplayNameA (Opnum 32)

The RGetServiceDisplayNameA method returns the display name of the specified service.

1. DWORD RGetServiceDisplayNameA(
2. [in] SC\_RPC\_HANDLE hSCManager,
3. [in, string, range(0, SC\_MAX\_NAME\_LENGTH)]
4. LPSTR lpServiceName,
5. [out, string, size\_is(\*lpcchBuffer)]
6. LPSTR lpDisplayName,
7. [in, out] LPBOUNDED\_DWORD\_4K lpcchBuffer
8. );

**hSCManager:** An [SC\_RPC\_HANDLE](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) (section 2.2.4) data type that defines the handle to the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282).

**lpServiceName:** A pointer to a null-terminated ANSI string that specifies the service name.

The forward slash, back slash, comma, and space characters are illegal in service names.

**lpDisplayName:** A pointer to a buffer that receives the null-terminated ANSI string that contains the service display name.

**lpcchBuffer:** An [LPBOUNDED\_DWORD\_4K](#Section_d5e2cc227db64101b36ff4377e836b4c) (section 2.2.7) data type that defines the pointer to a variable that specifies the size, in chars, of the buffer. On output, this variable receives the size of the service's display name, excluding the terminating null character.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 122  ERROR\_INSUFFICIENT\_BUFFER | The display name does not fit in the buffer. |
| 123  ERROR\_INVALID\_NAME | The specified service name is invalid. |
| 1060  ERROR\_SERVICE\_DOES\_NOT\_EXIST | The [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) with the specified ServiceName does not exist in the SCM database identified by the *hSCManager* parameter. |

In response to this request from the client, for a successful operation the server MUST look up the service record with the ServiceName matching the specified *lpServiceName* in the SCM database identified by the *hSCManager* parameter. The server MUST return the DisplayName from the found service record in the *lpDisplayName* parameter and set the size in chars of the display name excluding the terminating null character in *lpcchBuffer*.

If the *lpDisplayName* buffer is insufficient to hold the complete display name of the service, the server MUST fail the call with ERROR\_INSUFFICIENT\_BUFFER (122) and set the required size in chars of the display name excluding the terminating null character in *lpcchBuffer*.[<49>](#Appendix_A_49" \o "Product behavior note 49) If the size is sufficient for data returned, the server also returns the size that was set in *lpcchBuffer*.

If a service is created with a Unicode-encoded display name using the **RCreateServiceW** method, then the server MUST convert the display name to an ANSI string before returning it. The conversion process is specified in [[MS-UCODEREF]](%5bMS-UCODEREF%5d.pdf#Section_4a045e08fc294f22baf416f38c2825fb) section 3.1.5.1.1.2, Pseudocode for Mapping a UTF-16 String to an ANSI Codepage.

#### RGetServiceKeyNameA (Opnum 33)

The RGetServiceKeyNameA method returns the ServiceName of the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) with the specified DisplayName.

1. DWORD RGetServiceKeyNameA(
2. [in] SC\_RPC\_HANDLE hSCManager,
3. [in, string, range(0, SC\_MAX\_NAME\_LENGTH)]
4. LPSTR lpDisplayName,
5. [out, string, size\_is(\*lpcchBuffer)]
6. LPSTR lpKeyName,
7. [in, out] LPBOUNDED\_DWORD\_4K lpcchBuffer
8. );

**hSCManager:** An [SC\_RPC\_HANDLE](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) (section 2.2.4) data type that defines the handle to the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282).

**lpDisplayName:** A pointer to a null-terminated ANSI string that specifies the service display name.

**lpKeyName:** A pointer to a buffer that receives the null-terminated ANSI string that contains the service name.

**lpcchBuffer:** An [LPBOUNDED\_DWORD\_4K](#Section_d5e2cc227db64101b36ff4377e836b4c) (section 2.2.7) data type that defines the pointer to a variable that specifies the size, in chars, of the buffer. On output, this variable receives the size of the service name, excluding the terminating null character.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 122  ERROR\_INSUFFICIENT\_BUFFER | The data area passed to a system call is too small. |
| 123  ERROR\_INVALID\_NAME | The name specified in *lpDisplayName* is invalid or set to NULL. |
| 1060  ERROR\_SERVICE\_DOES\_NOT\_EXIST | The service record with the DisplayName matching the specified *lpDisplayName* does not exist in the SCM database identified by the *hSCManager* parameter. |

In response to this request from the client, for a successful operation the server MUST look up the service record with DisplayName matching the display name specified by the *lpDisplayName* parameter in the SCM database identified by *hSCManager*.

The server MUST return the ServiceName from the found service record in the *lpKeyName* parameter and set the size in chars of the service name excluding the terminating null character in *lpcchBuffer*.

If the *lpKeyName* buffer is insufficient to hold the complete service name of the service, the server MUST fail the call with ERROR\_INSUFFICIENT\_BUFFER (122) and set the required size in chars of the service name excluding the terminating null character in *lpcchBuffer*.[<50>](#Appendix_A_50" \o "Product behavior note 50) If the size is sufficient for data returned, the server also returns the size that was set in *lpcchBuffer*.

If a service record is created with a [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8)-encoded display name using the [RCreateServiceW](#Section_6a8ca92694774dd4b766692fab07227e) method, then the server MUST convert the service name to an [**ANSI**](#gt_100cd8a6-5cb1-4895-9de6-e4a3c224a583) string before returning it. The conversion process is specified in [[MS-UCODEREF]](%5bMS-UCODEREF%5d.pdf#Section_4a045e08fc294f22baf416f38c2825fb) section 3.1.5.1.1.2, Pseudocode for Mapping a UTF-16 String to an ANSI Codepage.

The server MUST use the process described in [Conversion Between ANSI and Unicode String Formats (section 3.1.7)](#Section_f9ddeebeb05e423c944084fd3d14bee9) to convert a string to the appropriate format.

#### REnumServiceGroupW (Opnum 35)

The REnumServiceGroupW method returns the members of a [**service group**](#gt_8ee1e5c0-3886-409f-8707-197e6232042d).

1. DWORD REnumServiceGroupW(
2. [in] SC\_RPC\_HANDLE hSCManager,
3. [in] DWORD dwServiceType,
4. [in] DWORD dwServiceState,
5. [out, size\_is(cbBufSize)] LPBYTE lpBuffer,
6. [in, range(0, 1024\*256)] DWORD cbBufSize,
7. [out] LPBOUNDED\_DWORD\_256K pcbBytesNeeded,
8. [out] LPBOUNDED\_DWORD\_256K lpServicesReturned,
9. [in, out, unique] LPBOUNDED\_DWORD\_256K lpResumeIndex,
10. [in, string, unique, range(0, SC\_MAX\_NAME\_LENGTH)]
11. LPCWSTR pszGroupName
12. );

**hSCManager:** An [SC\_RPC\_HANDLE](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) (section 2.2.4) data type that defines the handle to the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) created using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SC\_MANAGER\_ENUMERATE\_SERVICE access right MUST have been granted to the caller when the [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) was created.

**dwServiceType:** A value that specifies the [**service records**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) to enumerate based on their Type. This MUST be one or a combination of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_KERNEL\_DRIVER  0x00000001 | A driver service. These are services that manage devices on the system. |
| SERVICE\_FILE\_SYSTEM\_DRIVER  0x00000002 | A file system driver service. These are services that manage file systems on the system. |
| SERVICE\_WIN32\_OWN\_PROCESS  0x00000010 | Service that runs in its own process. |
| SERVICE\_WIN32\_SHARE\_PROCESS  0x00000020 | Service that shares a process with other services. |

**dwServiceState:** A value that specifies the service records to enumerate based on their ServiceStatus.dwCurrentState. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_ACTIVE  0x00000001 | Enumerates service records with ServiceStatus.dwCurrentState values from the following: SERVICE\_START\_PENDING, SERVICE\_STOP\_PENDING, SERVICE\_RUNNING, SERVICE\_CONTINUE\_PENDING, SERVICE\_PAUSE\_PENDING, and SERVICE\_PAUSED. |
| SERVICE\_INACTIVE  0x00000002 | Enumerates service records with the ServiceStatus.dwCurrentState value SERVICE\_STOPPED. |
| SERVICE\_STATE\_ALL  0x00000003 | Enumerates service records with ServiceStatus.dwCurrentState values from the following: SERVICE\_START\_PENDING, SERVICE\_STOP\_PENDING, SERVICE\_RUNNING, SERVICE\_CONTINUE\_PENDING, SERVICE\_PAUSE\_PENDING, SERVICE\_PAUSED, and SERVICE\_STOPPED. |

**lpBuffer:** A pointer to an array of [ENUM\_SERVICE\_STATUSW](#Section_79f17326c1384fbc81705c11c4b3ca79) (section 2.2.11) structures that contain the name and service status information for each dependent service in the database.

**cbBufSize:** The size, in bytes, of the array pointed to by *lpBuffer*.

**pcbBytesNeeded:** An [LPBOUNDED\_DWORD\_256K](#Section_6c441817d7364a94a4440c1810a6c473) (section 2.2.9) pointer to a variable that contains the number of bytes needed to store the array of service entries.

**lpServicesReturned:** An LPBOUNDED\_DWORD\_256K (section 2.2.9) pointer to a variable that contains the number of service entries returned.

**lpResumeIndex:** An LPBOUNDED\_DWORD\_256K (section 2.2.9) pointer to a variable that specifies the current position in the status enumeration. The server MUST assign a unique number to each service for the boot session, in increasing order, and increment that number by one for each service addition. The value of the *lpResumeIndex* parameter is one of these numbers, which the server can use to determine the resumption point for the enumeration.

**pszGroupName:** A pointer to a string that specifies service records to enumerate based on their ServiceGroup value.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SC\_MANAGER\_ENUMERATE\_SERVICE access right had not been granted to the caller when the RPC context handle was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 234  ERROR\_MORE\_DATA | More data is available. |
| 1060  ERROR\_SERVICE\_DOES\_NOT\_EXIST | The group specified by *pszGroupName* does not exist in the SCM GroupList. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST determine the list of service records in the SCM database identified by the *hSCManager* parameter with a ServiceGroup value matching the *pszGroupName* parameter, determine that their ServiceStatus.dwCurrentState is equal to the state specified by *dwServiceParameter*, and determine that their Type value is equal to the *dwServiceType* parameter of the client request. The server MUST return this list by setting the service name and state of each service in this list in the array of ENUM\_SERVICE\_STATUSW (section 2.2.11) structures pointed to by the *lpBuffer* parameter and MUST set number of services returned in the *lpServicesReturned* parameter.

The client MUST set *lpResumeIndex* to 0 on the first call. If the server fails the call with ERROR\_MORE\_DATA (234), then the server MUST return a non-zero value in *lpResumeIndex* that the client MUST then specify in the subsequent calls. The server MUST set this parameter to zero if the operation succeeds. If the *lpResumeIndex* value is set by the client to any non-zero number not returned by the server, the behavior is not defined.

If the size of the *lpServices* array is insufficient for the list of services returned, the server MUST fail the call with ERROR\_MORE\_DATA (234) and return the size, in bytes, required in the *pcbBytesNeeded* parameter. If the size is sufficient for data returned, the server also returns the required size, in bytes.

If the size of the *lpServices* array is sufficient for the list of services returned, the enumerated data MAY be in the buffer in a non-contiguous manner, and portions of the *lpServices* array MAY be empty (filled with 0x00).

The server MUST return ERROR\_INVALID\_PARAMETER (87) if a bitmask specified in *dwServiceState* is zero or contains undefined values.

The server MUST return ERROR\_INVALID\_PARAMETER (87) if a bitmask specified in *dwServiceType* is zero or contains undefined values.

#### RChangeServiceConfig2A (Opnum 36)

The RChangeServiceConfig2A method SHOULD[<51>](#Appendix_A_51" \o "Product behavior note 51) change the optional configuration parameters of a service.

1. DWORD RChangeServiceConfig2A(
2. [in] SC\_RPC\_HANDLE hService,
3. [in] SC\_RPC\_CONFIG\_INFOA Info
4. );

**hService:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SERVICE\_CHANGE\_CONFIG access right MUST have been granted to the caller when the [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) to the service record was created.

**Info:** An [SC\_RPC\_CONFIG\_INFOA (section 2.2.21)](#Section_f8c400d3328b4e6b9d858135c8f790a4) structure that contains optional configuration information.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise it returns one of the following error codes.[<52>](#Appendix_A_52" \o "Product behavior note 52)

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SERVICE\_CHANGE\_CONFIG access right had not been granted to the caller when the RPC context handle to the service record was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 1072  ERROR\_SERVICE\_MARKED\_FOR\_DELETE | The RDeleteService has been called for the service record identified by the *hService* parameter. |
| 1080  ERROR\_CANNOT\_DETECT\_DRIVER\_FAILURE | SERVICE\_CONFIG\_FAILURE\_ACTIONS cannot be used as a **dwInfoLevel** in the *Info* parameter for service records with a Type value defined for drivers. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST update the specific attributes of the service record identified by *hService*, using the information level and the corresponding values associated with that information level as specified in the *Info* parameter of the client request.

If the service has a PreferredNode setting and the client requested a change of a service record with a Type other than SERVICE\_WIN32\_OWN\_PROCESS, the server MUST fail the call with ERROR\_INVALID\_PARAMETER (87).

If the service record ServiceGroup value is set and the client specifies a start type of delayed autostart (see section [2.2.33](#Section_805b8296863d4d1e8ae8f639adf8c6cb)), the server MUST fail the call with ERROR\_INVALID\_PARAMETER (87).

The server MUST use the process described in [Conversion Between ANSI and Unicode String Formats (section 3.1.7)](#Section_f9ddeebeb05e423c944084fd3d14bee9) to convert a string to the appropriate format.

#### RChangeServiceConfig2W (Opnum 37)

The RChangeServiceConfig2W [<53>](#Appendix_A_53" \o "Product behavior note 53) method changes the optional configuration parameters of a service.

1. DWORD RChangeServiceConfig2W(
2. [in] SC\_RPC\_HANDLE hService,
3. [in] SC\_RPC\_CONFIG\_INFOW Info
4. );

**hService:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SERVICE\_CHANGE\_CONFIG access right MUST have been granted to the caller when the [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) to the service record was created.

**Info:** An [SC\_RPC\_CONFIG\_INFOW (section 2.2.22)](#Section_4225730329d24ea6b4d28d5a95e4e3e0) structure that contains optional configuration information.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise it returns one of the following error codes.[<54>](#Appendix_A_54" \o "Product behavior note 54)

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SERVICE\_CHANGE\_CONFIG access right had not been granted to the caller when the RPC context handle to the service record was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 1072  ERROR\_SERVICE\_MARKED\_FOR\_DELETE | The RDeleteService has been called for the service record identified by the *hService* parameter. |
| 1080  ERROR\_CANNOT\_DETECT\_DRIVER\_FAILURE | SERVICE\_CONFIG\_FAILURE\_ACTIONS cannot be used as a **dwInfoLevel** in the *Info* parameter for service records with a Type value defined for drivers. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST update the specific attributes of the service record identified by *hService*, using the information level and the corresponding values associated with that information level as specified in the *Info* parameter of the client request.

If the service has a PreferredNode setting and the client requested a change of a service record with a Type value other than SERVICE\_WIN32\_OWN\_PROCESS, the server MUST fail the call with ERROR\_INVALID\_PARAMETER (87).

If the service record ServiceGroup value is set and the client specifies a start type of delayed autostart (see section [2.2.33](#Section_805b8296863d4d1e8ae8f639adf8c6cb)), the server MUST fail the call with ERROR\_INVALID\_PARAMETER (87).

#### RQueryServiceConfig2A (Opnum 38)

The RQueryServiceConfig2A [<55>](#Appendix_A_55" \o "Product behavior note 55) method returns the optional configuration parameters of the specified service based on the specified information level.

1. DWORD RQueryServiceConfig2A(
2. [in] SC\_RPC\_HANDLE hService,
3. [in] DWORD dwInfoLevel,
4. [out, size\_is(cbBufSize)] LPBYTE lpBuffer,
5. [in, range(0, 1024\*8)] DWORD cbBufSize,
6. [out] LPBOUNDED\_DWORD\_8K pcbBytesNeeded
7. );

**hService:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SERVICE\_QUERY\_CONFIG access right MUST have been granted to the caller when the [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) to the service record was created.

**dwInfoLevel:** A value that specifies the configuration information to query. This SHOULD be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_CONFIG\_DESCRIPTION  0x00000001 | The *lpBuffer* parameter is a pointer to a [SERVICE\_DESCRIPTIONA](#Section_91a48e76b6b843eb9ea10757c4aa9260) structure. |
| SERVICE\_CONFIG\_FAILURE\_ACTIONS  0x00000002 | The *lpBuffer* parameter is a pointer to a [SERVICE\_FAILURE\_ACTIONSA](#Section_a5b207c7114e48aea1c6ef79c1fe1854) structure. |
| SERVICE\_CONFIG\_DELAYED\_AUTO\_START\_INFO  0x00000003[<56>](#Appendix_A_56" \o "Product behavior note 56) | The *lpBuffer* parameter is a pointer to a [SERVICE\_DELAYED\_AUTO\_START\_INFO](#Section_805b8296863d4d1e8ae8f639adf8c6cb) structure. |
| SERVICE\_CONFIG\_FAILURE\_ACTIONS\_FLAG  0x00000004[<57>](#Appendix_A_57" \o "Product behavior note 57) | The *lpBuffer* parameter is a pointer to a [SERVICE\_FAILURE\_ACTIONS\_FLAG](#Section_9b244e2e82fc4c548f4fb19034faa2c4) structure. |
| SERVICE\_CONFIG\_SERVICE\_SID\_INFO  0x00000005[<58>](#Appendix_A_58" \o "Product behavior note 58) | The *lpBuffer* parameter is a pointer to a [SERVICE\_SID\_INFO](#Section_ea1a9acd4bb2473fae5b55969c5960fc) structure. |
| SERVICE\_CONFIG\_REQUIRED\_PRIVILEGES\_INFO  0x00000006[<59>](#Appendix_A_59" \o "Product behavior note 59) | The *lpBuffer* parameter is a pointer to a [SERVICE\_RPC\_REQUIRED\_PRIVILEGES\_INFO](#Section_2917121dbbe147d2ad557bce44a127e6) structure. |
| SERVICE\_CONFIG\_PRESHUTDOWN\_INFO  0x00000007[<60>](#Appendix_A_60" \o "Product behavior note 60) | The *lpBuffer* parameter is a pointer to a [SERVICE\_PRESHUTDOWN\_INFO](#Section_6b55ad95df3e498fb107e77f831ba467) structure. |
| SERVICE\_CONFIG\_PREFERRED\_NODE  0x00000009[<61>](#Appendix_A_61" \o "Product behavior note 61) | The *lpBuffer* parameter is a pointer to a [SERVICE\_PREFERRED\_NODE\_INFO](#Section_3767259ff0544076acd39f7ecb79cab3) structure.[<62>](#Appendix_A_62" \o "Product behavior note 62) |

**lpBuffer:** A pointer to the buffer that contains the service configuration information. The format of this data depends on the value of the *dwInfoLevel* parameter.

**cbBufSize:** The size, in bytes, of the *lpBuffer* parameter.

**pcbBytesNeeded:** An [LPBOUNDED\_DWORD\_8K (section 2.2.8)](#Section_1720ed6ed90741c68d98fb6e4877d761) data type that defines the pointer to a variable that contains the number of bytes needed to return the configuration information.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SERVICE\_QUERY\_CONFIG access right had not been granted to the caller when the RPC context handle to the service record was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 122  ERROR\_INSUFFICIENT\_BUFFER | The data area passed to a system call is too small. |
| 124  ERROR\_INVALID\_LEVEL | The *dwInfoLevel* parameter contains an unsupported value. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST query the specific configuration information stored in the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database associated with the service record identified by the *hService* parameter, using the information level and the corresponding values associated with that information level as specified in the *dwInfoLevel* parameter of the client request. The server MUST return this configuration data by setting the *lpBuffer* parameter with the appropriate structure filled with the configuration data based on *dwInfoLevel*.

The server MUST set the required buffer size in the *pcbBytesNeeded* parameter.

If the buffer pointed to by *lpBuffer* is insufficient to hold all the configuration data, the server MUST fail the call with ERROR\_INSUFFICIENT\_BUFFER (122).

The server MUST use the process described in [Conversion Between ANSI and Unicode String Formats (section 3.1.7)](#Section_f9ddeebeb05e423c944084fd3d14bee9) to convert a string to the appropriate format.

The server MUST return ERROR\_INVALID\_PARAMETER (87) if either or both *lpBuffer* and *pcbBytesNeeded* are NULL.[<63>](#Appendix_A_63" \o "Product behavior note 63)

#### RQueryServiceConfig2W (Opnum 39)

The RQueryServiceConfig2W [<64>](#Appendix_A_64" \o "Product behavior note 64) method returns the optional configuration parameters of the specified service based on the specified information level.

1. DWORD RQueryServiceConfig2W(
2. [in] SC\_RPC\_HANDLE hService,
3. [in] DWORD dwInfoLevel,
4. [out, size\_is(cbBufSize)] LPBYTE lpBuffer,
5. [in, range(0, 1024\*8)] DWORD cbBufSize,
6. [out] LPBOUNDED\_DWORD\_8K pcbBytesNeeded
7. );

**hService:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SERVICE\_QUERY\_CONFIG access right MUST have been granted to the caller when the [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) to the service record was created.

| Value | Meaning |
| --- | --- |
| SERVICE\_CONFIG\_DESCRIPTION  0x00000001 | The *lpBuffer* parameter is a pointer to a [SERVICE\_DESCRIPTION\_WOW64 (section 2.2.36)](#Section_90b40ea4c24f42a281503b5173bfc90a) structure. |
| SERVICE\_CONFIG\_FAILURE\_ACTIONS  0x00000002 | The *lpBuffer* parameter is a pointer to a [SERVICE\_FAILURE\_ACTIONS\_WOW64 (section 2.2.37)](#Section_9489030edd8249db8505b3d72c6a9271) structure. |
| SERVICE\_CONFIG\_DELAYED\_AUTO\_START\_INFO  0x00000003[<65>](#Appendix_A_65" \o "Product behavior note 65) | The *lpBuffer* parameter is a pointer to a [SERVICE\_DELAYED\_AUTO\_START\_INFO](#Section_805b8296863d4d1e8ae8f639adf8c6cb) structure. |
| SERVICE\_CONFIG\_FAILURE\_ACTIONS\_FLAG  0x00000004[<66>](#Appendix_A_66" \o "Product behavior note 66) | The *lpBuffer* parameter is a pointer to a [SERVICE\_FAILURE\_ACTIONS\_FLAG](#Section_9b244e2e82fc4c548f4fb19034faa2c4) structure. |
| SERVICE\_CONFIG\_SERVICE\_SID\_INFO  0x00000005[<67>](#Appendix_A_67" \o "Product behavior note 67) | The *lpBuffer* parameter is a pointer to a [SERVICE\_SID\_INFO](#Section_ea1a9acd4bb2473fae5b55969c5960fc) structure. |
| SERVICE\_CONFIG\_REQUIRED\_PRIVILEGES\_INFO  0x00000006[<68>](#Appendix_A_68" \o "Product behavior note 68) | The *lpBuffer* parameter is a pointer to a [SERVICE\_REQUIRED\_PRIVILEGES\_INFO\_WOW64 (section 2.2.38)](#Section_99ecc0e7b2d1466c8e227f2b71786ff6) structure. |
| SERVICE\_CONFIG\_PRESHUTDOWN\_INFO  0x00000007[<69>](#Appendix_A_69" \o "Product behavior note 69) | The *lpBuffer* parameter is a pointer to a [SERVICE\_PRESHUTDOWN\_INFO](#Section_6b55ad95df3e498fb107e77f831ba467) structure. |
| SERVICE\_CONFIG\_PREFERRED\_NODE  0x00000009[<70>](#Appendix_A_70" \o "Product behavior note 70) | The *lpInfo* parameter is a pointer to a [SERVICE\_PREFERRED\_NODE\_INFO](#Section_3767259ff0544076acd39f7ecb79cab3) structure.[<71>](#Appendix_A_71" \o "Product behavior note 71) |

**dwInfoLevel:** A value that specifies the configuration information to query. This MUST be one of the following values.

**lpBuffer:** A pointer to the buffer that contains the service configuration information. The format of this data depends on the value of the *dwInfoLevel* parameter.

When *dwInfoLevel* is SERVICE\_CONFIG\_DESCRIPTION, or SERVICE\_CONFIG\_FAILURE\_ACTIONS or SERVICE\_CONFIG\_REQUIRED\_PRIVILEGES\_INFO, the server returns an *lpBuffer* parameter that has the requested data and the offset to the start of the data from the top of the buffer. The API converts the offset into pointers that it returns to the caller by means of the *lpBuffer* parameter.

**cbBufSize:** The size, in bytes, of the *lpBuffer* parameter.

**pcbBytesNeeded:** An [LPBOUNDED\_DWORD\_8K (section 2.2.8)](#Section_1720ed6ed90741c68d98fb6e4877d761) data type that defines the pointer to a variable that receives the number of bytes needed to return the configuration information.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The SERVICE\_QUERY\_CONFIG access right had not been granted to the caller when the RPC context handle to the service record was created. |
| 0x00000006  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 0x00000087  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 0x00000122  ERROR\_INSUFFICIENT\_BUFFER | The data area passed to a system call is too small. |
| 0x00000124  ERROR\_INVALID\_LEVEL | The *dwInfoLevel* parameter contains an unsupported value. |
| 0x00001115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST query the specific configuration information stored in the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database associated with the service record identified by the *hService* parameter, using the information level and the corresponding values associated with that information level as specified in the *dwInfoLevel* parameter of the client request. The server MUST return this configuration data by setting the *lpBuffer* parameter with the appropriate structure filled with the configuration data based on *dwInfoLevel*.

The server MUST set the required buffer size in the *pcbBytesNeeded* parameter.

If the buffer pointed to by *lpBuffer* is insufficient to hold all the configuration data, the server MUST fail the call with ERROR\_INSUFFICIENT\_BUFFER (122).

The server MUST return ERROR\_INVALID\_PARAMETER (87) if either or both *lpBuffer* and *pcbBytesNeeded* are NULL.[<72>](#Appendix_A_72" \o "Product behavior note 72)

#### RQueryServiceStatusEx (Opnum 40)

The RQueryServiceStatusEx method returns the current status of the specified service, based on the specified information level.

1. DWORD RQueryServiceStatusEx(
2. [in] SC\_RPC\_HANDLE hService,
3. [in] SC\_STATUS\_TYPE InfoLevel,
4. [out, size\_is(cbBufSize)] LPBYTE lpBuffer,
5. [in, range(0, 1024\*8)] DWORD cbBufSize,
6. [out] LPBOUNDED\_DWORD\_8K pcbBytesNeeded
7. );

**hService:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SERVICE\_QUERY\_STATUS access right MUST have been granted to the caller when the [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) to the service record was created.

**InfoLevel:** An enumerated value from [SC\_STATUS\_TYPE (section 2.2.29)](#Section_a7de3a4b0b9e4b9b8863b3dbc9bbe02b) that specifies which service attributes are returned. MUST be SC\_STATUS\_PROCESS\_INFO.

**lpBuffer:** A pointer to the buffer that contains the status information in the form of a [SERVICE\_STATUS\_PROCESS (section 2.2.49)](#Section_c2f0ab87eb1643718380ddf9cd29931e) structure.

**cbBufSize:** The size, in bytes, of the *lpBuffer* parameter.

**pcbBytesNeeded:** An [LPBOUNDED\_DWORD\_8K (section 2.2.8)](#Section_1720ed6ed90741c68d98fb6e4877d761) data type that defines the pointer to a variable that contains the number of bytes needed to return the configuration information.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SERVICE\_QUERY\_STATUS access right had not been granted to the caller when the RPC context handle to the service record was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 122  ERROR\_INSUFFICIENT\_BUFFER | The data area passed to a system call is too small. |
| 124  ERROR\_INVALID\_LEVEL | The *InfoLevel* parameter contains an unsupported value. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST query the configuration information as specified and stored in the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database associated with the service record identified by the *hService* parameter. The server MUST return this configuration data by setting the *lpBuffer* parameter with the SERVICE\_STATUS\_PROCESS structure filled with the configuration data as specified in section 2.2.49.

If the buffer pointed to by *lpBuffer* is insufficient to hold all the configuration data, the server MUST fail the call with ERROR\_INSUFFICIENT\_BUFFER (122) and set the required buffer size in the *pcbBytesNeeded* parameter.

#### REnumServicesStatusExA (Opnum 41)

The REnumServicesStatusExA method enumerates services in the specified [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database, based on the specified information level.

1. DWORD REnumServicesStatusExA(
2. [in] SC\_RPC\_HANDLE hSCManager,
3. [in] SC\_ENUM\_TYPE InfoLevel,
4. [in] DWORD dwServiceType,
5. [in] DWORD dwServiceState,
6. [out, size\_is(cbBufSize)] LPBYTE lpBuffer,
7. [in, range(0, 1024 \* 256)] DWORD cbBufSize,
8. [out] LPBOUNDED\_DWORD\_256K pcbBytesNeeded,
9. [out] LPBOUNDED\_DWORD\_256K lpServicesReturned,
10. [in, out, unique] LPBOUNDED\_DWORD\_256K lpResumeIndex,
11. [in, string, unique, range(0, SC\_MAX\_NAME\_LENGTH)]
12. LPCSTR pszGroupName
13. );

**hSCManager:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the SCM database that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SC\_MANAGER\_ENUMERATE\_SERVICE access right MUST have been granted to the caller when the [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) to the SCM was created.

**InfoLevel:** An [SC\_ENUM\_TYPE (section 2.2.20)](#Section_9ee28cc619b7464b8464d645e4189e76) structure that specifies which service attributes to return. MUST be SC\_ENUM\_PROCESS\_INFO.

**dwServiceType:** A value that specifies what type of [**service records**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) to enumerate. This MUST be one or a combination of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_KERNEL\_DRIVER  0x0000000F | Enumerates services of type SERVICE\_KERNEL\_DRIVER. |
| SERVICE\_FILE\_SYSTEM\_DRIVER  0x00000002 | Enumerates services of type SERVICE\_FILE\_SYSTEM\_DRIVER. |
| SERVICE\_WIN32\_OWN\_PROCESS  0x00000010 | Enumerates services of type SERVICE\_WIN32\_OWN\_PROCESS. |
| SERVICE\_WIN32\_SHARE\_PROCESS  0x00000020 | Enumerates services of type SERVICE\_WIN32\_SHARE\_PROCESS. |

**dwServiceState:** Value that specifies the service records to enumerate based on their **ServiceStatus.dwCurrentState**. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_ACTIVE  0x00000001 | Enumerates service records with **ServiceStatus.dwCurrentState** values from the following: SERVICE\_START\_PENDING, SERVICE\_STOP\_PENDING, SERVICE\_RUNNING, SERVICE\_CONTINUE\_PENDING, SERVICE\_PAUSE\_PENDING, and SERVICE\_PAUSED. |
| SERVICE\_INACTIVE  0x00000002 | Enumerates service records with the **ServiceStatus.dwCurrentState** value SERVICE\_STOPPED. |
| SERVICE\_STATE\_ALL  0x00000003 | Enumerates service records with **ServiceStatus.dwCurrentState** values from the following: SERVICE\_START\_PENDING, SERVICE\_STOP\_PENDING, SERVICE\_RUNNING, SERVICE\_CONTINUE\_PENDING, SERVICE\_PAUSE\_PENDING, SERVICE\_PAUSED, and SERVICE\_STOPPED. |

**lpBuffer:** A pointer to the buffer that contains the status information in the form of an array of [ENUM\_SERVICE\_STATUS\_PROCESSA (section 2.2.12)](#Section_45beeb69d09d427c945b9e8a774a0d09) structures.

**cbBufSize:** The size, in bytes, of the buffer pointed to by *lpBuffer*.

**pcbBytesNeeded:** An [LPBOUNDED\_DWORD\_256K (section 2.2.9)](#Section_6c441817d7364a94a4440c1810a6c473) pointer to a variable that contains the number of bytes needed to return the configuration information.

**lpServicesReturned:** An LPBOUNDED\_DWORD\_256K (section 2.2.9) pointer to a variable that contains the number of service entries returned.

**lpResumeIndex:** An LPBOUNDED\_DWORD\_256K (section 2.2.9) pointer to a variable that contains the current index in the enumerated list of service entries. The server MUST assign a unique number to each service for the boot session, in increasing order, and increment that number by one for each service addition. The value of the *lpResumeIndex* parameter is one of these numbers, which the server can use to determine the resumption point for the enumeration.

**pszGroupName:** A pointer to a string that specifies service records to enumerate based on their **ServiceGroup** values.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SC\_MANAGER\_ENUMERATE\_SERVICE access right had not been granted to the caller when the RPC context handle to the SCM was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 124  ERROR\_INVALID\_LEVEL | The *InfoLevel* parameter contains an unsupported value. |
| 234  ERROR\_MORE\_DATA | More data is available. |
| 1060  ERROR\_SERVICE\_DOES\_NOT\_EXIST | The group specified by the *pszGroupName* parameter does not exist in the SCM GroupList. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST determine the list of service records in the SCM database identified by the *hSCManager* parameter with the **ServiceGroup** value matching the *pszGroupName* parameter, the **ServiceStatus.dwCurrentState** equal to the state specified by *dwServiceState*, and the **Type** equal to *dwServiceType* of the client request. The server MUST return this list by setting the service name, display name, and appropriate configuration data for each of the services in the list in the array of ENUM\_SERVICE\_STATUS\_PROCESSA (section 2.2.12) structures pointed to by the *lpBuffer* parameter and MUST set the number of services returned in the *lpServicesReturned* parameter.

If the *lpResumeIndex* value is not zero, the server MUST use that as the offset to the service list and return only services starting at this offset. If the *lpResumeIndex* value is zero, the server MUST return all services. The server MUST set this parameter to zero if the operation succeeds. If the *lpResumeIndex* value is set by the client to any nonzero number not returned by the server, the behavior is not defined.

If the *pszGroupName* parameter is a nonempty or non-NULL string, the server MUST enumerate only the services that belong to the group whose name is specified by the *pszGroupName* parameter. If the *pszGroupName* parameter is an empty string, the server MUST enumerate only the services that do not belong to any group. If the *pszGroupName* parameter is NULL, the server MUST ignore the group membership and enumerate all services.

If the size of the *lpBuffer* array is insufficient for the list of services returned, the server MUST fail the call with ERROR\_MORE\_DATA (234) and return the size in bytes required in the *pcbBytesNeeded* parameter. If the size is sufficient for data returned, the server also returns the required size, in bytes.

If the size of the *lpBuffer* array is sufficient for the list of services returned, the enumerated data MAY be in the buffer in a non-contiguous manner, and portions of the *lpBuffer* array MAY be empty.

The server MUST use the process described in [Conversion Between ANSI and Unicode String Formats (section 3.1.7)](#Section_f9ddeebeb05e423c944084fd3d14bee9) to convert a string to the appropriate format.

The server MUST return ERROR\_INVALID\_PARAMETER (87) if a bitmask specified in *dwServiceState* is zero or contains undefined values.

The server MUST return ERROR\_INVALID\_PARAMETER (87) if a bitmask specified in *dwServiceType* is zero or contains undefined values.

#### REnumServicesStatusExW (Opnum 42)

The REnumServicesStatusExW method enumerates services in the specified [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database, based on the specified information level.

1. DWORD REnumServicesStatusExW(
2. [in] SC\_RPC\_HANDLE hSCManager,
3. [in] SC\_ENUM\_TYPE InfoLevel,
4. [in] DWORD dwServiceType,
5. [in] DWORD dwServiceState,
6. [out, size\_is(cbBufSize)] LPBYTE lpBuffer,
7. [in, range(0, 1024\*256)] DWORD cbBufSize,
8. [out] LPBOUNDED\_DWORD\_256K pcbBytesNeeded,
9. [out] LPBOUNDED\_DWORD\_256K lpServicesReturned,
10. [in, out, unique] LPBOUNDED\_DWORD\_256K lpResumeIndex,
11. [in, string, unique, range(0, SC\_MAX\_NAME\_LENGTH)]
12. LPCWSTR pszGroupName
13. );

**hSCManager:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the SCM database that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SC\_MANAGER\_ENUMERATE\_SERVICE access right MUST have been granted to the caller when the [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) to the SCM was created.

**InfoLevel:** An [SC\_ENUM\_TYPE (section 2.2.20)](#Section_9ee28cc619b7464b8464d645e4189e76) structure that specifies which service attributes are returned. This MUST be SC\_ENUM\_PROCESS\_INFO.

**dwServiceType:** A value that specifies the [**service records**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) to enumerate based on their Type. This MUST be one or a combination of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_KERNEL\_DRIVER  0x00000001 | Enumerates services of type SERVICE\_KERNEL\_DRIVER. |
| SERVICE\_FILE\_SYSTEM\_DRIVER  0x00000002 | Enumerates services of type SERVICE\_FILE\_SYSTEM\_DRIVER. |
| SERVICE\_WIN32\_OWN\_PROCESS  0x00000010 | Enumerates services of type SERVICE\_WIN32\_OWN\_PROCESS. |
| SERVICE\_WIN32\_SHARE\_PROCESS  0x00000020 | Enumerates services of type SERVICE\_WIN32\_SHARE\_PROCESS. |

**dwServiceState:** A value that specifies the service records to enumerate based on their **ServiceStatus.dwCurrentState**. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_ACTIVE  0x00000001 | Enumerates service records with **ServiceStatus.dwCurrentState** values from the following: SERVICE\_START\_PENDING, SERVICE\_STOP\_PENDING, SERVICE\_RUNNING, SERVICE\_CONTINUE\_PENDING, SERVICE\_PAUSE\_PENDING, and SERVICE\_PAUSED. |
| SERVICE\_INACTIVE  0x00000002 | Enumerates service records with the **ServiceStatus.dwCurrentState** value SERVICE\_STOPPED. |
| SERVICE\_STATE\_ALL  0x00000003 | Enumerates service records with **ServiceStatus.dwCurrentState** values from the following: SERVICE\_START\_PENDING, SERVICE\_STOP\_PENDING, SERVICE\_RUNNING, SERVICE\_CONTINUE\_PENDING, SERVICE\_PAUSE\_PENDING, SERVICE\_PAUSED, and SERVICE\_STOPPED. |

**lpBuffer:** A pointer to the buffer that contains the status information in the form of an array of [ENUM\_SERVICE\_STATUS\_PROCESSW (section 2.2.13)](#Section_fe73e0f37ed148eb8265a6f8c33e3e05) structures.

**cbBufSize:** The size, in bytes, of the buffer pointed to by *lpBuffer*.

**pcbBytesNeeded:** An [LPBOUNDED\_DWORD\_256K (section 2.2.9)](#Section_6c441817d7364a94a4440c1810a6c473) pointer to a variable that contains the number of bytes needed to return the configuration information if the method fails.

**lpServicesReturned:** An LPBOUNDED\_DWORD\_256K (section 2.2.9) pointer to a variable that contains the number of service entries returned.

**lpResumeIndex:** An LPBOUNDED\_DWORD\_256K (section 2.2.9) pointer to a variable that contains the current index in the enumerated list of service entries. The server MUST assign a unique number to each service for the boot session, in increasing order, and increment that number by one for each service addition. The value of the *lpResumeIndex* parameter is one of these numbers, which the server can use to determine the resumption point for the enumeration.

**pszGroupName:** A pointer to a string that specifies service records to enumerate based on their **ServiceGroup** values.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SC\_MANAGER\_ENUMERATE\_SERVICE access right had not been granted to the caller when the RPC context handle to the SCM was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 124  ERROR\_INVALID\_LEVEL | The *InfoLevel* parameter contains an unsupported value. |
| 234  ERROR\_MORE\_DATA | More data is available. |
| 1060  ERROR\_SERVICE\_DOES\_NOT\_EXIST | The group specified by the *pszGroupName* parameter does not exist in the SCM GroupList. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST determine the list of service records in the SCM database identified by the *hSCManager* parameter with a **ServiceGroup** value matching the *pszGroupName* parameter, **ServiceStatus.dwCurrentState** equal to the state specified by *dwServiceState*, and **Type** equal to *dwServiceType* of the client request. The server MUST return this list by setting the service name, display name, and the appropriate configuration data for each of the services in the list in the array of ENUM\_SERVICE\_STATUS\_PROCESSW (section 2.2.13) structures pointed to by the *lpBuffer* parameter and MUST set the number of services returned in the *lpServicesReturned* parameter.

If the *lpResumeIndex* value is not zero, the server MUST use that as the offset to the service list and return only services starting at this offset. If the lpResumeIndex value is zero, the server MUST return all services. The server MUST set this parameter to zero if the operation succeeds. If the *lpResumeIndex* value is set by the client to any nonzero number not returned by the server, the behavior is not defined.

If the *pszGroupName* parameter is a nonempty or non-NULL string, the server MUST enumerate only the services that belong to the group whose name is specified by the *pszGroupName* parameter. If the *pszGroupName* parameter is an empty string, the server MUST enumerate only the services that do not belong to any group. If the *pszGroupName* parameter is NULL, the server MUST ignore the group membership and enumerate all services.

If the size of the *lpBuffer* array is insufficient for the list of services returned, the server MUST fail the call with ERROR\_MORE\_DATA (234) and return the size in bytes required in the *pcbBytesNeeded* parameter. If the size is sufficient for data returned, the server also returns the required size, in bytes.

If the size of the *lpBuffer* array is sufficient for the list of services returned, the enumerated data MAY be in the buffer in a non-contiguous manner, and portions of the *lpBuffer* array MAY be empty.

The server MUST return ERROR\_INVALID\_PARAMETER (87) if a bitmask specified in *dwServiceState* is zero or contains undefined values.

The server MUST return ERROR\_INVALID\_PARAMETER (87) if a bitmask specified in *dwServiceType* is zero or contains undefined values.

#### RCreateServiceWOW64A (Opnum 44)

The RCreateServiceWOW64A method creates the service record for a 32-bit service on a 64-bit system with the path to the file image automatically adjusted to point to a 32-bit file location on the system.

1. DWORD RCreateServiceWOW64A(
2. [in] SC\_RPC\_HANDLE hSCManager,
3. [in, string, range(0, SC\_MAX\_NAME\_LENGTH)]
4. LPSTR lpServiceName,
5. [in, string, unique, range(0, SC\_MAX\_NAME\_LENGTH)]
6. LPSTR lpDisplayName,
7. [in] DWORD dwDesiredAccess,
8. [in] DWORD dwServiceType,
9. [in] DWORD dwStartType,
10. [in] DWORD dwErrorControl,
11. [in, string, range(0, SC\_MAX\_PATH\_LENGTH)]
12. LPSTR lpBinaryPathName,
13. [in, string, unique, range(0, SC\_MAX\_NAME\_LENGTH)]
14. LPSTR lpLoadOrderGroup,
15. [in, out, unique] LPDWORD lpdwTagId,
16. [in, unique, size\_is(dwDependSize)]
17. LPBYTE lpDependencies,
18. [in, range(0, SC\_MAX\_DEPEND\_SIZE)]
19. DWORD dwDependSize,
20. [in, string, unique, range(0, SC\_MAX\_ACCOUNT\_NAME\_LENGTH)]
21. LPSTR lpServiceStartName,
22. [in, unique, size\_is(dwPwSize)]
23. LPBYTE lpPassword,
24. [in, range(0, SC\_MAX\_PWD\_SIZE)]
25. DWORD dwPwSize,
26. [out] LPSC\_RPC\_HANDLE lpServiceHandle
27. );

**hSCManager:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SC\_MANAGER\_CREATE\_SERVICE access right MUST have been granted to the caller when the [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) to the SCM was created.

**lpServiceName:** A pointer to a null-terminated ANSI string that specifies the name of the service to install. This MUST not be null.

**lpDisplayName:** A pointer to a null-terminated ANSI string that contains the display name by which user interface programs identify the service.

**dwDesiredAccess:** A value that specifies the access to the service. This MUST be one of the values as specified in section 3.1.4.

**dwServiceType:** A value that specifies the type of service. This MUST be one or a combination of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_KERNEL\_DRIVER  0x00000001 | A driver service. These are services that manage devices on the system. |
| SERVICE\_FILE\_SYSTEM\_DRIVER  0x00000002 | A file system driver service. These are services that manage file systems on the system. |
| SERVICE\_WIN32\_OWN\_PROCESS  0x00000010 | Service that runs within its own process. |
| SERVICE\_WIN32\_SHARE\_PROCESS  0x00000020 | Service that shares an execution process with other services. |
| SERVICE\_INTERACTIVE\_PROCESS  0x00000100 | The service can interact with the desktop. |

**dwStartType:** A value that specifies when to start the service. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_BOOT\_START  0x00000000 | Starts the driver service when the system boots up. This value is valid only for driver services. |
| SERVICE\_SYSTEM\_START  0x00000001 | Starts the driver service when the system boots up. This value is valid only for driver services. The services marked SERVICE\_SYSTEM\_START are started after all SERVICE\_BOOT\_START services have been started. |
| SERVICE\_AUTO\_START  0x00000002 | Starts the service automatically during system startup. |
| SERVICE\_DEMAND\_START  0x00000003 | Starts the service when a client requests the SCM to start the service. |
| SERVICE\_DISABLED  0x00000004 | Service cannot be started. |

**dwErrorControl:** A value that specifies the severity of the error if the service fails to start and determines the action that the SCM takes. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_ERROR\_IGNORE  0x00000000 | The SCM ignores the error and continues the startup operation. |
| SERVICE\_ERROR\_NORMAL  0x00000001 | The SCM logs the error, but continues the startup operation. |
| SERVICE\_ERROR\_SEVERE  0x00000002 | The SCM 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  0x00000003 | The SCM SHOULD log 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. |

**lpBinaryPathName:** A pointer to a null-terminated ANSI string that contains the fully qualified path to the service binary file. The path MAY include arguments. If the path contains a space, it MUST be quoted so that it is correctly interpreted. For example, "d:\\my share\\myservice.exe" is specified as "\"d:\\my share\\myservice.exe\"".

**lpLoadOrderGroup:** A pointer to a null-terminated ANSI string that names the load-ordering group of which this service is a member.

Specify NULL or an empty string if the service does not belong to a [**load-ordering group**](#gt_88c26bf0-5b6f-4423-82b3-1027ea5df0e3).

**lpdwTagId:** A pointer to a variable that receives a tag value. The value is unique to the group specified in the *lpLoadOrderGroup* parameter.

**lpDependencies:** A pointer to an array of null-separated names of services or load ordering groups that MUST start before this service. The array is doubly null-terminated. Load ordering group names are prefixed with a "+" character (to distinguish them from service names). If the pointer is NULL or if it points to an empty string, the service has no dependencies. Cyclic dependency between services is not allowed. The character set is [**ANSI**](#gt_100cd8a6-5cb1-4895-9de6-e4a3c224a583). 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.

**dwDependSize:** The size, in bytes, of the string specified by the *dwDependSize* parameter.

**lpServiceStartName:** A pointer to a null-terminated ANSI that specifies the name of the account under which the service runs.

**lpPassword:** A pointer to a null-terminated ANSI string that contains the password of the account whose name was specified by the *lpServiceStartName* parameter.

**dwPwSize:** The size, in bytes, of the password specified by the *lpPassword* parameter.

**lpServiceHandle:** An LPSC\_RPC\_HANDLE (section 2.2.4) data type that defines the handle to the newly created [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721).

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SC\_MANAGER\_CREATE\_SERVICE access right had not been granted to the caller when the RPC context handle to the SCM was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle specified is invalid. |
| 13  ERROR\_INVALID\_DATA | The data is invalid. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 123  ERROR\_INVALID\_NAME | The specified service name is invalid. |
| 1057  ERROR\_INVALID\_SERVICE\_ACCOUNT | The user account name specified in the *lpServiceStartName* parameter does not exist. |
| 1059  ERROR\_CIRCULAR\_DEPENDENCY | A circular service dependency was specified. |
| 1072  ERROR\_SERVICE\_MARKED\_FOR\_DELETE | The service record with a specified name already exists and RDeleteService has been called for it. |
| 1073  ERROR\_SERVICE\_EXISTS | The service record with the ServiceName matching the specified *lpServiceName* already exists. |
| 1078  ERROR\_DUPLICATE\_SERVICE\_NAME | The service record with the same DisplayName or the same ServiceName as the passed-in *lpDisplayName* already exists in the SCM database. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST use the service name specified in the *lpServiceName* parameter to create a new service record in the SCM database and use the values from the appropriate parameters of the client request to update the attributes of this newly created service record.

The only valid combinations of values for *dwServiceType* are SERVICE\_INTERACTIVE\_PROCESS and SERVICE\_WIN32\_OWN\_PROCESS or SERVICE\_INTERACTIVE\_PROCESS and SERVICE\_WIN32\_SHARE\_PROCESS. If the value of *dwServiceType* has more than one bit set and the combination of bits is not equal to SERVICE\_INTERACTIVE\_PROCESS and SERVICE\_WIN32\_OWN\_PROCESS or SERVICE\_INTERACTIVE\_PROCESS and SERVICE\_WIN32\_SHARE\_PROCESS, the server MUST fail the method and return the error ERROR\_INVALID\_PARAMETER.

The server MUST convert the location specified in the *lpBinaryPathName* parameter to point to the 32-bit location on a 64-bit system.[<73>](#Appendix_A_73" \o "Product behavior note 73)

If the service is created successfully, the server MUST return a handle to the service in the *lpServiceHandle* parameter with the access rights associated with this handle as specified in the *dwDesiredAccess* parameter of the client request.

The server MUST use the process described in [Conversion Between ANSI and Unicode String Formats (section 3.1.7)](#Section_f9ddeebeb05e423c944084fd3d14bee9) to convert a string to the appropriate format.

If *lpBinaryPathName* contains arguments, the server MUST pass these arguments to the service entry point.

If *lpdwTagId* has a valid value and *lpLoadOrderGroup* is either NULL or an empty string, then the server MUST return ERROR\_INVALID\_PARAMETER.

#### RCreateServiceWOW64W (Opnum 45)

The RCreateServiceWOW64W method creates the service record for a 32-bit service on a 64-bit system with the path to the file image automatically adjusted to point to a 32-bit file location on the system.

1. DWORD RCreateServiceWOW64W(
2. [in] SC\_RPC\_HANDLE hSCManager,
3. [in, string, range(0, SC\_MAX\_NAME\_LENGTH)]
4. wchar\_t\* lpServiceName,
5. [in, string, unique, range(0, SC\_MAX\_NAME\_LENGTH)]
6. wchar\_t\* lpDisplayName,
7. [in] DWORD dwDesiredAccess,
8. [in] DWORD dwServiceType,
9. [in] DWORD dwStartType,
10. [in] DWORD dwErrorControl,
11. [in, string, range(0, SC\_MAX\_PATH\_LENGTH)]
12. wchar\_t\* lpBinaryPathName,
13. [in, string, unique, range(0, SC\_MAX\_NAME\_LENGTH)]
14. wchar\_t\* lpLoadOrderGroup,
15. [in, out, unique] LPDWORD lpdwTagId,
16. [in, unique, size\_is(dwDependSize)]
17. LPBYTE lpDependencies,
18. [in, range(0, SC\_MAX\_DEPEND\_SIZE)]
19. DWORD dwDependSize,
20. [in, string, unique, range(0, SC\_MAX\_ACCOUNT\_NAME\_LENGTH)]
21. wchar\_t\* lpServiceStartName,
22. [in, unique, size\_is(dwPwSize)]
23. LPBYTE lpPassword,
24. [in, range(0, SC\_MAX\_PWD\_SIZE)]
25. DWORD dwPwSize,
26. [out] LPSC\_RPC\_HANDLE lpServiceHandle
27. );

**hSCManager:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database created using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SC\_MANAGER\_CREATE\_SERVICE access right MUST have been granted to the caller when the [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) to the SCM was created.

**lpServiceName:** A pointer to a null-terminated **UNICODE** string that specifies the name of the service to install. This MUST NOT be NULL.

The forward slash, back slash, comma, and space characters are illegal in service names.

**lpDisplayName:** A pointer to a null-terminated **UNICODE** string that contains the display name by which user interface programs identify the service.

**dwDesiredAccess:** A value that specifies the access to the service. This MUST be one of the values as specified in section 3.1.4.

**dwServiceType:** A value that specifies the type of service. This MUST be one or a combination of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_KERNEL\_DRIVER  0x00000001 | A driver service. These are services that manage devices on the system. |
| SERVICE\_FILE\_SYSTEM\_DRIVER  0x00000002 | A file system driver service. These are services that manage file systems on the system. |
| SERVICE\_WIN32\_OWN\_PROCESS  0x00000010 | Service that runs within its own process. |
| SERVICE\_WIN32\_SHARE\_PROCESS  0x00000020 | Service that shares a process with other services. |
| SERVICE\_INTERACTIVE\_PROCESS  0x00000100 | The service can interact with the desktop. |

**dwStartType:** A value that specifies when to start the service. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_BOOT\_START  0x00000000 | Starts the driver service when the system boots up. This value is valid only for driver services. |
| SERVICE\_SYSTEM\_START  0x00000001 | Starts the driver service when the system boots up. This value is valid only for driver services. The services marked SERVICE\_SYSTEM\_START are started after all SERVICE\_BOOT\_START services have been started. |
| SERVICE\_AUTO\_START  0x00000002 | Starts the service automatically during system startup. |
| SERVICE\_DEMAND\_START  0x00000003 | Starts the service when a client requests the SCM to start the service. |
| SERVICE\_DISABLED  0x00000004 | Service cannot be started. |

**dwErrorControl:** A value that specifies the severity of the error if the service fails to start and determines the action that the SCM takes. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_ERROR\_IGNORE  0x00000000 | The SCM ignores the error and continues the startup operation. |
| SERVICE\_ERROR\_NORMAL  0x00000001 | The SCM logs the error, but continues the startup operation. |
| SERVICE\_ERROR\_SEVERE  0x00000002 | The SCM 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  0x00000003 | The SCM SHOULD log 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. |

**lpBinaryPathName:** A pointer to a null-terminated UNICODE string that contains the fully qualified path to the service binary file. The path MAY include arguments. If the path contains a space, it MUST be quoted so that it is correctly interpreted. For example, "d:\\my share\\myservice.exe" is specified as "\"d:\\my share\\myservice.exe\"".

**lpLoadOrderGroup:** A pointer to a null-terminated UNICODE string that names the load-ordering group of which this service is a member.

Specify NULL or an empty string if the service does not belong to a [**load-ordering group**](#gt_88c26bf0-5b6f-4423-82b3-1027ea5df0e3).

**lpdwTagId:** A pointer to a variable that receives a tag value. The value is unique to the group specified in the *lpLoadOrderGroup* parameter.

**lpDependencies:** A pointer to an array of null-separated names of services or load ordering groups that MUST start before this service. The array is doubly null-terminated. Load ordering group names are prefixed with a "+" character (to distinguish them from service names). If the pointer is NULL or if it points to an empty string, the service has no dependencies. Cyclic dependency between services is not allowed. The character set is Unicode. 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.

**dwDependSize:** The size, in bytes, of the string specified by the *dwDependSize* parameter.

**lpServiceStartName:** A pointer to a null-terminated UNICODE string that specifies the name of the account under which the service runs.

**lpPassword:** A pointer to a null-terminated UNICODE string that contains the password of the account whose name was specified by the *lpServiceStartName* parameter.

**dwPwSize:** The size, in bytes, of the password specified by the *lpPassword* parameter.

**lpServiceHandle:** An LPSC\_RPC\_HANDLE (section 2.2.4) data type that defines the handle to the newly created [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721).

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SC\_MANAGER\_CREATE\_SERVICE access right had not been granted to the caller when the RPC context handle to the SCM was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle specified is invalid. |
| 13  ERROR\_INVALID\_DATA | The data is invalid. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 123  ERROR\_INVALID\_NAME | The specified service name is invalid. |
| 1057  ERROR\_INVALID\_SERVICE\_ACCOUNT | The user account name specified in the *lpServiceStartName* parameter does not exist. |
| 1059  ERROR\_CIRCULAR\_DEPENDENCY | A circular service dependency was specified. |
| 1072  ERROR\_SERVICE\_MARKED\_FOR\_DELETE | The service record with a specified name already exists, and RDeleteService has been called for it. |
| 1073  ERROR\_SERVICE\_EXISTS | The service record with the ServiceName matching the specified *lpServiceName* already exists. |
| 1078  ERROR\_DUPLICATE\_SERVICE\_NAME | The service record with the same DisplayName or the same ServiceName as the passed-in *lpDisplayName* already exists in the service control manager database. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST use the service name specified in the *lpServiceName* parameter to create a new service record in the SCM database and use the values from the appropriate parameters of the client request to update the attributes of this newly created service record.

The server MUST convert the location specified in the *lpBinaryPathName* parameter to point to the 32-bit location on a 64-bit system.

If the service is created successfully, the server MUST return a handle to the service in the *lpServiceHandle* parameter with the access rights associated with this handle as specified in the *dwDesiredAccess* parameter of the client request.

The only valid combinations of values for *dwServiceType* are SERVICE\_INTERACTIVE\_PROCESS and SERVICE\_WIN32\_OWN\_PROCESS or SERVICE\_INTERACTIVE\_PROCESS and SERVICE\_WIN32\_SHARE\_PROCESS. If the value of *dwServiceType* has more than one bit set and the combination of bits is not equal to SERVICE\_INTERACTIVE\_PROCESS and SERVICE\_WIN32\_OWN\_PROCESS or SERVICE\_INTERACTIVE\_PROCESS and SERVICE\_WIN32\_SHARE\_PROCESS, the server MUST fail the method and return the error ERROR\_INVALID\_PARAMETER.

If *lpBinaryPathName* contains arguments, the server MUST pass these arguments to the service entry point.

If *lpdwTagId* has a valid value and *lpLoadOrderGroup* is either NULL or an empty string, then the server MUST return ERROR\_INVALID\_PARAMETER.

#### RNotifyServiceStatusChange (Opnum 47)

The RNotifyServiceStatusChange method[<74>](#Appendix_A_74" \o "Product behavior note 74) allows the client to register for notifications and check, via [RGetNotifyResults (section 3.1.4.44)](#Section_7020f3e117b9495680dc583ea1509fe6), when the specified service of type SERVICE\_WIN32\_OWN\_PROCESS or SERVICE\_WIN32\_SHARE\_PROCESS is created or deleted or when its status changes.

1. DWORD RNotifyServiceStatusChange(
2. [in] SC\_RPC\_HANDLE hService,
3. [in] SC\_RPC\_NOTIFY\_PARAMS NotifyParams,
4. [in] GUID\* pClientProcessGuid,
5. [out] GUID\* pSCMProcessGuid,
6. [out] PBOOL pfCreateRemoteQueue,
7. [out] LPSC\_NOTIFY\_RPC\_HANDLE phNotify
8. );

**hService:** An [SC\_RPC\_HANDLE](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the SCM for SERVICE\_NOTIFY\_CREATED and SERVICE\_NOTIFY\_DELETED notifications or to the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) for all other notification types that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SC\_MANAGER\_ENUMERATE\_SERVICE access right MUST have been granted to the caller when the [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) to the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) was created, or the SERVICE\_QUERY\_STATUS access right MUST have been granted to the caller when the RPC context handle to the service record was created.

**NotifyParams:** An [SC\_RPC\_NOTIFY\_PARAMS](#Section_0e88b912da92411d915ab9b7df6d64fc) (section 2.2.23) data type that defines the service status notification information.

**pClientProcessGuid:** Not used. This MUST be ignored.

**pSCMProcessGuid:** Not used. This MUST be ignored.

**pfCreateRemoteQueue:** Not used. This MUST be ignored.

**phNotify:** An [LPSC\_NOTIFY\_RPC\_HANDLE](#Section_40e0fec2d70a41829a45b5da7e2d0943) (section 2.2.6) data type that defines a handle to the notification status associated with the client for the specified service.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SC\_MANAGER\_ENUMERATE\_SERVICE access right had not been granted to the caller when the RPC context handle to the SCM was created, or the SERVICE\_QUERY\_STATUS access right had not been granted to the caller when the RPC context handle to the service record was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid or is not supported for the specified notification. |
| 50  ERROR\_NOT\_SUPPORTED | The request is not supported. |
| 87  ERROR\_INVALID\_PARAMETER | A parameter that was specified is invalid. |
| 124  ERROR\_INVALID\_LEVEL | The system call level is not correct. |
| 1072  ERROR\_SERVICE\_MARKED\_FOR\_DELETE | The RDeleteService has been called for the service record identified by the *hService* parameter. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |
| 1242  ERROR\_ALREADY\_REGISTERED | A notification status handle has already been created for the service handle passed in the *hService* parameter. |
| 1294  ERROR\_SERVICE\_NOTIFY\_CLIENT\_LAGGING | The service notification client is lagging too far behind the current state of services in the machine. |

In response to this request from the client, for a successful operation, the server MUST associate NOTIFY\_RPC\_HANDLE for the caller to check for status changes using RGetNotifyResults for the service record identified by the *hService* parameter.

The server MUST ignore any value set in the *ullThreadId* parameter in *NotifyParams*.

The server MUST fail the call and return ERROR\_INVALID\_PARAMETER if *dwNotifyMask* contains masks for both create/delete events and service status events.

The client can set the value of *pClientProcessGuid*, *pSCMProcessGuid*, and *pfCreatRemoteQueue* to any value, such as 0, and the server MUST ignore these.

The server MUST return ERROR\_NOT\_SUPPORTED (50) if the value of *dwInfoLevel* is greater than SERVICE\_NOTIFY\_STATUS\_CHANGE.

The server MUST return ERROR\_INVALID\_LEVEL (124) if the value of *dwInfoLevel* is not SERVICE\_NOTIFY\_STATUS\_CHANGE (0x2) or SERVICE\_NOTIFY\_STATUS\_CHANGE\_1 (0x1).

#### RGetNotifyResults (Opnum 48)

The RGetNotifyResults method[<75>](#Appendix_A_75" \o "Product behavior note 75) returns notification information when the specified status change that was previously requested by the client via [RNotifyServiceStatusChange (section 3.1.4.43)](#Section_b4cb24471f8c4deea78e209bdacadea6) occurs on a specified service.

The client MUST make one call to RGetNotifyResults for each call to RNotifyServiceStatusChange.

1. error\_status\_t RGetNotifyResults(
2. [in] SC\_NOTIFY\_RPC\_HANDLE hNotify,
3. [out] PSC\_RPC\_NOTIFY\_PARAMS\_LIST\* ppNotifyParams
4. );

**hNotify:** An [SC\_NOTIFY\_RPC\_HANDLE (section 2.2.6)](#Section_40e0fec2d70a41829a45b5da7e2d0943) data type that defines a handle to the notification status associated with the client. This is the handle returned by an RNotifyServiceStatusChange call.

**ppNotifyParams:** A pointer to a buffer that receives an [SC\_RPC\_NOTIFY\_PARAMS\_LIST (section 2.2.24)](#Section_46063c6407cb4055985c75fffb259f41) data type.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |
| 1235  ERROR\_REQUEST\_ABORTED | The request was aborted. |

In response to this request, the server MUST wait until the service state in the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) changes to one of the values specified in the [SC\_RPC\_NOTIFY\_PARAMS (section 2.2.23)](#Section_0e88b912da92411d915ab9b7df6d64fc) structure passed to the **RNotifyServiceStatusChange** method that returned the *hNotify* parameter. When the service changes state to one of the values specified in the SC\_RPC\_NOTIFY\_PARAMS structure associated with the *hNotify* parameter, the server MUST update the client by setting the appropriate values in the *ppNotifyParams* parameter and returning the call.[<76>](#Appendix_A_76" \o "Product behavior note 76)

The client MUST ignore any value set in the *ullThreadId* parameter in *ppNotifyParams*.

#### RCloseNotifyHandle (Opnum 49)

The RCloseNotifyHandle method[<77>](#Appendix_A_77" \o "Product behavior note 77) unregisters the client from receiving future notifications via the [RGetNotifyResults (section 3.1.4.44)](#Section_7020f3e117b9495680dc583ea1509fe6) method from the server for specified status changes on a specified service.

1. DWORD RCloseNotifyHandle(
2. [in, out] LPSC\_NOTIFY\_RPC\_HANDLE phNotify,
3. [out] PBOOL pfApcFired
4. );

**phNotify:** An [SC\_NOTIFY\_RPC\_HANDLE (section 2.2.6)](#Section_40e0fec2d70a41829a45b5da7e2d0943) data type that defines a handle to the notification status associated with the client. This is the handle returned by an [RNotifyServiceStatusChange](#Section_b4cb24471f8c4deea78e209bdacadea6) call.

**pfApcFired:** Not used.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns the following error code.

| Return value/code | Description |
| --- | --- |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |

In response to this request from the client, for a successful operation the server MUST close the handle specified in the *phNotify* parameter and stop notifying the client about status changes for the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) associated with the handle.

#### RControlServiceExA (Opnum 50)

The RControlServiceExA method[<78>](#Appendix_A_78" \o "Product behavior note 78) receives a control code for a specific service.

1. DWORD RControlServiceExA(
2. [in] SC\_RPC\_HANDLE hService,
3. [in] DWORD dwControl,
4. [in] DWORD dwInfoLevel,
5. [in, switch\_is(dwInfoLevel)] PSC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSA pControlInParams,
6. [out, switch\_is(dwInfoLevel)] PSC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSA pControlOutParams
7. );

**hService:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the service record that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282).

**dwControl:** Requested control code. This MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_CONTROL\_STOP  0x00000001 | Notifies a service to stop. The SERVICE\_STOP access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_STOP bit set in the **ServiceStatus.dwControlsAccepted** field of the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721). |
| SERVICE\_CONTROL\_PAUSE  0x00000002 | Notifies a service to pause. The SERVICE\_PAUSE\_CONTINUE access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_PAUSE\_CONTINUE bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |
| SERVICE\_CONTROL\_CONTINUE  0x00000003 | Notifies a paused service to resume. The SERVICE\_PAUSE\_CONTINUE access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_PAUSE\_CONTINUE bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |
| SERVICE\_CONTROL\_INTERROGATE  0x00000004 | Notifies a service to report its current status information to the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84). The SERVICE\_INTERROGATE access right MUST have been granted to the caller when the RPC control handle to the service record was created. |
| SERVICE\_CONTROL\_PARAMCHANGE  0x00000006 | Notifies a service that its startup parameters have changed. The SERVICE\_PAUSE\_CONTINUE access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_PARAMCHANGE bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |
| SERVICE\_CONTROL\_NETBINDADD  0x00000007 | Notifies a service that there is a new component for binding. The SERVICE\_PAUSE\_CONTINUE access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_NETBINDCHANGE bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |
| SERVICE\_CONTROL\_NETBINDREMOVE  0x00000008 | Notifies a network service that a component for binding has been removed. The SERVICE\_PAUSE\_CONTINUE access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_NETBINDCHANGE bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |
| SERVICE\_CONTROL\_NETBINDENABLE  0x00000009 | Notifies a network service that a disabled binding has been enabled. The SERVICE\_PAUSE\_CONTINUE access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_NETBINDCHANGE bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |
| SERVICE\_CONTROL\_NETBINDDISABLE  0x0000000A | Notifies a network service that one of its bindings has been disabled. The SERVICE\_PAUSE\_CONTINUE access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_NETBINDCHANGE bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |

Services can define their own codes in the range 128-255.

**dwInfoLevel:** The information level for the service control parameters. This MUST be set to 0x00000001.

**pControlInParams:** A pointer to a [SERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSA (section 2.2.30)](#Section_003621fc656f41be9368324587f3904a) structure that contains the reason associated with the SERVICE\_CONTROL\_STOP control.

**pControlOutParams:** A pointer to a buffer that contains a [SERVICE\_CONTROL\_STATUS\_REASON\_OUT\_PARAMS (section 2.2.32)](#Section_2679fdcc5e6a40928958a625f2a4ace7) structure to receive the current status on the service.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The required access right had not been granted to the caller when the RPC context handle to the service record was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 87  ERROR\_INVALID\_PARAMETER | The requested control code is undefined. |
| 124  ERROR\_INVALID\_LEVEL | The *dwInfoLevel* parameter contains an unsupported value. |
| 1051  ERROR\_DEPENDENT\_SERVICES\_RUNNING | The service cannot be stopped because other running services are dependent on it. |
| 1052  ERROR\_INVALID\_SERVICE\_CONTROL | The requested control code is not valid, or it is unacceptable to the service. |
| 1053  ERROR\_SERVICE\_REQUEST\_TIMEOUT | The process for the service was started, but it did not respond within an implementation-specific time-out.[<79>](#Appendix_A_79" \o "Product behavior note 79) |
| 1061  ERROR\_SERVICE\_CANNOT\_ACCEPT\_CTRL | The requested control code cannot be sent to the service because the state of the service is **SERVICE\_START\_PENDING** or **SERVICE\_STOP\_PENDING**. |
| 1062  ERROR\_SERVICE\_NOT\_ACTIVE | The service has not been started, or the **ServiceStatus.dwCurrentState** in the service record is **SERVICE\_STOPPED**. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the SCM MUST send the control specified in the *dwControl* parameter to the service created for the service record identified by the *hService* parameter of the client request if the type of the service record is SERVICE\_WIN32\_OWN\_PROCESS or SERVICE\_WIN32\_SHARE\_PROCESS.

If the type of the service record is SERVICE\_KERNEL\_DRIVER or SERVICE\_FILESYSTEM\_DRIVER, and *dwControl* parameter is not SERVICE\_CONTROL\_INTERROGATE or SERVICE\_CONTROL\_STOP, the SCM MUST fail the request with ERROR\_INVALID\_SERVICE\_CONTROL.

If the type of the service record is SERVICE\_KERNEL\_DRIVER or SERVICE\_FILESYSTEM\_DRIVER, the SCM MUST query the current status of the driver from the operating system and set the **ServiceStatus.dwCurrentState** of the service record to SERVICE\_RUNNING if driver is loaded and to SERVICE\_STOPPED if it is not.

If the *dwControl* is not SERVICE\_CONTROL\_INTERROGATE and the type of the service record is SERVICE\_KERNEL\_DRIVER or SERVICE\_FILESYSTEM\_DRIVER and the driver is managed by the PnP subsystem, the SCM MUST fail the request with ERROR\_INVALID\_SERVICE\_CONTROL. In response to this request from the client, for a successful operation the SCM MUST return the current status of the service by setting *pControlOutParams* after the operation.

If the **ServiceStatus.dwControlsAccepted** field of the service record does not have a required SERVICE\_ACCEPT\_xxx bit set, the SCM MUST fail the request with ERROR\_INVALID\_SERVICE\_CONTROL.

If the *dwInfoLevel* parameter of the client request is set to 0x00000001, the server MUST provide information in *pControlOutParams*.

The server MUST return the services last known state if *dwControl* is SERVICE\_CONTROL\_INTERROGATE and the service is in START\_PENDING state.

If *dwControl* is not equal to SERVICE\_CONTROL\_STOP, pControlInParams->pszComment MUST be NULL. If not, the server MUST fail the call and return ERROR\_INVALID\_PARAMETER (87).

The server MUST use the process described in [Conversion Between ANSI and Unicode String Formats (section 3.1.7)](#Section_f9ddeebeb05e423c944084fd3d14bee9) to convert a string to the appropriate format.

#### RControlServiceExW (Opnum 51)

The RControlServiceExW method[<80>](#Appendix_A_80" \o "Product behavior note 80) receives a control code for a specific service.

1. DWORD RControlServiceExW(
2. [in] SC\_RPC\_HANDLE hService,
3. [in] DWORD dwControl,
4. [in] DWORD dwInfoLevel,
5. [in, switch\_is(dwInfoLevel)] PSC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSW pControlInParams,
6. [out, switch\_is(dwInfoLevel)] PSC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSW pControlOutParams
7. );

**hService:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the service record that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282).

**dwControl:** Requested control code. MUST be one of the following values.

| Value | Meaning |
| --- | --- |
| SERVICE\_CONTROL\_STOP  0x00000001 | Notifies a service to stop. The SERVICE\_STOP access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_STOP bit set in the **ServiceStatus.dwControlsAccepted** field of the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721). |
| SERVICE\_CONTROL\_PAUSE  0x00000002 | Notifies a service to pause. The SERVICE\_PAUSE\_CONTINUE access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_PAUSE\_CONTINUE bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |
| SERVICE\_CONTROL\_CONTINUE  0x00000003 | Notifies a paused service to resume. The SERVICE\_PAUSE\_CONTINUE access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_PAUSE\_CONTINUE bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |
| SERVICE\_CONTROL\_INTERROGATE  0x00000004 | Notifies a service to report its current status information to the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84). The SERVICE\_INTERROGATE access right MUST have been granted to the caller when the RPC control handle to the service record was created. |
| SERVICE\_CONTROL\_PARAMCHANGE  0x00000006 | Notifies a service that its startup parameters have changed. The SERVICE\_PAUSE\_CONTINUE access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_PARAMCHANGE bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |
| SERVICE\_CONTROL\_NETBINDADD  0x00000007 | Notifies a service that there is a new component for binding. The SERVICE\_PAUSE\_CONTINUE access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_NETBINDCHANGE bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |
| SERVICE\_CONTROL\_NETBINDREMOVE  0x00000008 | Notifies a network service that a component for binding has been removed. The SERVICE\_PAUSE\_CONTINUE access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_NETBINDCHANGE bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |
| SERVICE\_CONTROL\_NETBINDENABLE  0x00000009 | Notifies a network service that a disabled binding has been enabled. The SERVICE\_PAUSE\_CONTINUE access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_NETBINDCHANGE bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |
| SERVICE\_CONTROL\_NETBINDDISABLE  0x0000000A | Notifies a network service that one of its bindings has been disabled. The SERVICE\_PAUSE\_CONTINUE access right MUST have been granted to the caller when the RPC control handle to the service record was created. The service record MUST have the SERVICE\_ACCEPT\_NETBINDCHANGE bit set in the **ServiceStatus.dwControlsAccepted** field of the service record. |

Services can define their own codes in the range 128-255.

**dwInfoLevel:** The information level for the service control parameters. This MUST be set to 0x00000001.

**pControlInParams:** A pointer to a [SERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSW (section 2.2.31)](#Section_1ac2684aea98420cac337441af91282e) structure that contains the reason associated with the SERVICE\_CONTROL\_STOP control.

**pControlOutParams:** A pointer to a buffer that contains a [SERVICE\_CONTROL\_STATUS\_REASON\_OUT\_PARAMS (section 2.2.32)](#Section_2679fdcc5e6a40928958a625f2a4ace7) structure to receive the current status on the service.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The required access right had not been granted to the caller when the RPC context handle to the service record was created. |
| 0x00000006  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 0x00000087  ERROR\_INVALID\_PARAMETER | The requested control code is undefined. |
| 0x00000124  ERROR\_INVALID\_LEVEL | The *dwInfoLevel* parameter contains an unsupported level. |
| 0x00001051  ERROR\_DEPENDENT\_SERVICES\_RUNNING | The service cannot be stopped because other running services are dependent on it. |
| 0x00001052  ERROR\_INVALID\_SERVICE\_CONTROL | The requested control code is not valid, or it is unacceptable to the service. |
| 0x00001053  ERROR\_SERVICE\_REQUEST\_TIMEOUT | The process for the service was started, but it did not respond within an implementation-specific timeout.[<81>](#Appendix_A_81" \o "Product behavior note 81) |
| 0x00001061  ERROR\_SERVICE\_CANNOT\_ACCEPT\_CTRL | The requested control code cannot be sent to the service because the state of the service is **SERVICE\_START\_PENDING** or **SERVICE\_STOP\_PENDING**. |
| 0x00001062  ERROR\_SERVICE\_NOT\_ACTIVE | The service has not been started, or the **ServiceStatus.dwCurrentState** in the service record is **SERVICE\_STOPPED**. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the SCM MUST send the control specified in the *dwControl* parameter to the service created for the service record identified by the *hService* parameter of the client request if the type of the service record is SERVICE\_WIN32\_OWN\_PROCESS or SERVICE\_WIN32\_SHARE\_PROCESS.

If the type of the service record is SERVICE\_KERNEL\_DRIVER or SERVICE\_FILESYSTEM\_DRIVER, and *dwControl* parameter is not SERVICE\_CONTROL\_INTERROGATE or SERVICE\_CONTROL\_STOP, the SCM MUST fail the request with ERROR\_INVALID\_SERVICE\_CONTROL.

If the type of the service record is SERVICE\_KERNEL\_DRIVER or SERVICE\_FILESYSTEM\_DRIVER, the SCM MUST query the current status of the driver from the Operating System and set the **ServiceStatus.dwCurrentState** of the service record to SERVICE\_RUNNING if driver is loaded and SERVICE\_STOPPED if it is not.

If the *dwControl* is not SERVICE\_CONTROL\_INTERROGATE and type of the service record is SERVICE\_KERNEL\_DRIVER or SERVICE\_FILESYSTEM\_DRIVER and the driver is managed by the PnP subsystem, the SCM MUST fail the request with ERROR\_INVALID\_SERVICE\_CONTROL.

If the **ServiceStatus.dwControlsAccepted** field of the service record does not have a required SERVICE\_ACCEPT\_xxx bit set, the SCM MUST fail the request with ERROR\_INVALID\_SERVICE\_CONTROL.

In response to this request from the client, for a successful operation the SCM MUST return the current status of the service by setting *pControlOutParams* after the operation.

The server MUST return the services last known state if *dwControl* is SERVICE\_CONTROL\_INTERROGATE and the service is in START\_PENDING state.

The server MUST provide information in *pControlOutParams*.

If *dwControl* is not equal to SERVICE\_CONTROL\_STOP, pControlInParams->pszComment MUST be NULL. If not, the server MUST fail the call and return ERROR\_INVALID\_PARAMETER (87).

#### RQueryServiceConfigEx (Opnum 56)

The RQueryServiceConfigEx method SHOULD[<82>](#Appendix_A_82" \o "Product behavior note 82) query the optional configuration parameters of a service.

1. DWORD RQueryServiceConfigEx(
2. [in] SC\_RPC\_HANDLE hService,
3. [in] DWORD dwInfoLevel,
4. [out] SC\_RPC\_CONFIG\_INFOW\* pInfo
5. );

**hService:** An [SC\_RPC\_HANDLE (section 2.2.4)](#Section_b5f0a0a8887c4097af1c71ef9f214ce4) data type that defines the handle to the [**service record**](#gt_1fbb7936-8437-4e3d-b62f-47df3be07721) that MUST have been created previously, using one of the open methods specified in section [3.1.4](#Section_0d7a70119f41470dad528535b47ac282). The SERVICE\_QUERY\_CONFIG access right MUST have been granted to the caller when the [**RPC context handle**](#gt_9dfe84e3-a772-4ad1-9a25-1255c1dc0f84) was created.

**dwInfoLevel:** The information level for the service configuration parameters. This MUST be set to 0x00000008 which corresponds to the service's trigger information.

**pInfo:** A pointer to an [SC\_RPC\_CONFIG\_INFOW (section 2.2.22)](#Section_4225730329d24ea6b4d28d5a95e4e3e0) structure that contains optional configuration information.

**Return Values:** The method returns 0x00000000 (ERROR\_SUCCESS) on success; otherwise, it returns one of the following error codes.

| Return value/code | Description |
| --- | --- |
| 5  ERROR\_ACCESS\_DENIED | The SERVICE\_QUERY\_CONFIG access right had not been granted to the caller when the RPC context handle was created. |
| 6  ERROR\_INVALID\_HANDLE | The handle is no longer valid. |
| 124  ERROR\_INVALID\_LEVEL | The *dwInfoLevel* parameter contains an unsupported value. |
| 1115  ERROR\_SHUTDOWN\_IN\_PROGRESS | The system is shutting down. |

In response to this request from the client, for a successful operation the server MUST query the specific configuration information stored in the SCM database in the service record identified by the *hService* parameter, using the information level and the corresponding values associated with that information level as specified in the *dwInfoLevel* parameter of the client request. The server MUST return this configuration data by setting the *pInfo* parameter with the appropriate structure filled with the configuration data based on *dwInfoLevel*.

The server MUST return a service's trigger information by returning a SERVICE\_TRIGGER\_INFO structure.

### Timer Events

None.

### Other Local Events

None.

### Conversion Between ANSI and Unicode String Formats

For all methods that require conversion, the server utilizes the conversion process specified in [[MS-UCODEREF]](%5bMS-UCODEREF%5d.pdf#Section_4a045e08fc294f22baf416f38c2825fb) section 3.1.5.1.1.2.

## RPC Runtime Check Notes

The behavior of the client when methods are executed can be affected by the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) protocol runtime checks and [**MIDL**](#gt_9c5903c1-1477-4181-b451-3ba1e34a0c0c) compiler options used when generating stubs. For example, this often concerns error codes when passing the NULL value in parameters with the [string] [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) attribute. In these cases, the IDL method does not return the expected error code. Instead, an RPC exception is raised.

For more information about generating RPC stubs from IDL definitions, see the topic "Using the MIDL Compiler" in [[MSDN-MIDL]](https://go.microsoft.com/fwlink/?LinkId=90041).

# Protocol Examples

The client receives a request from an application such as Services.msc to open the [**SCM**](#gt_afc3f782-a908-486d-9c05-8a054cd9cc84) database on the server for reading. After establishing a connection to the server, the client sends an [ROpenSCManagerW](#Section_dc84adb3d51d48eb820dba1c6ca5faf2) call with the following values for the parameters.

1. lpMachineName = "Name of the Server"
2. lpDatabaseName = "ServicesActive"
3. dwDesiredAccess = 0x00000001
4. lpScHandle = NULL

Upon receiving this request from the client, the server opens the handle to the SCM database with read access, the method returns an error code of 0, and the pointer is set to the opened handle in the *lpScHandle* parameter of the response.

The client can then use the handle returned in *lpScHandle* to operate on SCM database. For instance, to query the display name associated with a service, the client sends an [RGetServiceDisplayNameW](#Section_48b42c4a26b74d599b7a98533ebeb730) call with the following values for the parameters.

1. hSCManager = Handle returned in the lpScHandle parameter of the
2. previous server response.
3. lpServiceName = "GenericService\0"
4. lpDisplayName = Pointer to buffer that will receive the display name
5. lpcchBuffer = Size of the buffer pointed to by the lpDisplayName
6. parameter

Upon receiving this request from the client, the server queries the display name associated with the service "GenericService", the method returns an error code of 0, and then the server fills the display name in the buffer pointed to by the *lpDisplayName* parameter of the response.

When it is finished operating on the SCM database, the client closes the handle to this database by sending an [RCloseServiceHandle](#Section_a2a4e17409fb4e55bad3f77c4b13245c) with the following values for the parameters.

1. hSCObject = Handle returned in the lpScHandle parameter of the server
2. response to the ROpenSCManagerW call.

Upon receiving this request from the client, the server closes the handle to the open SCM database, and the method returns an error code of 0.

# Security

The following sections specify security considerations for implementers of the Service Control Manager Remote Protocol.

## Security Considerations for Implementers

None.

## Index of Security Parameters

| Security parameter | Section |
| --- | --- |
| RPC\_C\_AUTHN\_GSS\_NEGOTIATE | [2.1](#Section_69bbbbfa6f2e449ea5bc47db341fe81b) |
| RPC\_C\_AUTHN\_WINNT | 2.1 |
| RPC\_C\_AUTHN\_LEVEL\_PKT\_PRIVACY | 2.1 |
| RPC\_C\_AUTHN\_LEVEL\_CONNECT | 2.1 |

# Appendix A: Full IDL

For ease of implementation, the full [**Interface Definition Language (IDL)**](#gt_73177eec-4092-420f-92c5-60b2478df824) is provided as follows, where "ms-dtyp.idl" is the IDL found in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) Appendix A.[<83>](#Appendix_A_83" \o "Product behavior note 83)

1. import "ms-dtyp.idl";
2. [
3. uuid(367ABB81-9844-35F1-AD32-98F038001003),
4. version(2.0),
5. ms\_union,
6. pointer\_default(unique)
7. ]
8. interface svcctl{
9. const unsigned int MAX\_SERVICE\_NAME\_LENGTH = 256;
10. const unsigned short SC\_MAX\_DEPEND\_SIZE = 4 \* 1024;
11. const unsigned short SC\_MAX\_NAME\_LENGTH = MAX\_SERVICE\_NAME\_LENGTH + 1;
12. const unsigned short SC\_MAX\_PATH\_LENGTH = 32 \* 1024;
13. const unsigned short SC\_MAX\_PWD\_SIZE = 514;
14. const unsigned short SC\_MAX\_COMPUTER\_NAME\_LENGTH = 1024;
15. const unsigned short SC\_MAX\_ACCOUNT\_NAME\_LENGTH = 2 \* 1024;
16. const unsigned short SC\_MAX\_COMMENT\_LENGTH = 128;
17. const unsigned short SC\_MAX\_ARGUMENT\_LENGTH = 1024;
18. const unsigned short SC\_MAX\_ARGUMENTS = 1024;
19. typedef [handle]
20. wchar\_t\* SVCCTL\_HANDLEW;
21. typedef [handle]
22. LPSTR SVCCTL\_HANDLEA;
23. typedef [context\_handle] PVOID SC\_RPC\_HANDLE;
24. typedef [context\_handle] PVOID SC\_RPC\_LOCK;
25. typedef [context\_handle] PVOID SC\_NOTIFY\_RPC\_HANDLE;
26. typedef SC\_RPC\_HANDLE \* LPSC\_RPC\_HANDLE;
27. typedef SC\_RPC\_LOCK \* LPSC\_RPC\_LOCK;
28. typedef SC\_NOTIFY\_RPC\_HANDLE \* LPSC\_NOTIFY\_RPC\_HANDLE;
29. typedef struct \_STRING\_PTRSA {
30. [string, range(0, SC\_MAX\_ARGUMENT\_LENGTH)] LPSTR StringPtr;
31. } STRING\_PTRSA, \*PSTRING\_PTRSA, \*LPSTRING\_PTRSA;
32. typedef struct \_STRING\_PTRSW {
33. [string, range(0, SC\_MAX\_ARGUMENT\_LENGTH)] wchar\_t\* StringPtr;
34. } STRING\_PTRSW, \*PSTRING\_PTRSW, \*LPSTRING\_PTRSW;
35. typedef [range(0, 1024 \* 4)] DWORD BOUNDED\_DWORD\_4K;
36. typedef BOUNDED\_DWORD\_4K \* LPBOUNDED\_DWORD\_4K;
37. typedef [range(0, 1024 \* 8)] DWORD BOUNDED\_DWORD\_8K;
38. typedef BOUNDED\_DWORD\_8K \* LPBOUNDED\_DWORD\_8K;
39. typedef [range(0, 1024 \* 256)] DWORD BOUNDED\_DWORD\_256K;
40. typedef BOUNDED\_DWORD\_256K \* LPBOUNDED\_DWORD\_256K;
41. typedef struct {
42. DWORD dwServiceType;
43. DWORD dwCurrentState;
44. DWORD dwControlsAccepted;
45. DWORD dwWin32ExitCode;
46. DWORD dwServiceSpecificExitCode;
47. DWORD dwCheckPoint;
48. DWORD dwWaitHint;
49. } SERVICE\_STATUS,
50. \*LPSERVICE\_STATUS;
51. typedef struct {
52. DWORD dwServiceType;
53. DWORD dwCurrentState;
54. DWORD dwControlsAccepted;
55. DWORD dwWin32ExitCode;
56. DWORD dwServiceSpecificExitCode;
57. DWORD dwCheckPoint;
58. DWORD dwWaitHint;
59. DWORD dwProcessId;
60. DWORD dwServiceFlags;
61. } SERVICE\_STATUS\_PROCESS,
62. \*LPSERVICE\_STATUS\_PROCESS;
63. typedef struct \_QUERY\_SERVICE\_CONFIGW {
64. DWORD dwServiceType;
65. DWORD dwStartType;
66. DWORD dwErrorControl;
67. [string,range(0, 8 \* 1024)] LPWSTR lpBinaryPathName;
68. [string,range(0, 8 \* 1024)] LPWSTR lpLoadOrderGroup;
69. DWORD dwTagId;
70. [string,range(0, 8 \* 1024)] LPWSTR lpDependencies;
71. [string,range(0, 8 \* 1024)] LPWSTR lpServiceStartName;
72. [string,range(0, 8 \* 1024)] LPWSTR lpDisplayName;
73. } QUERY\_SERVICE\_CONFIGW,
74. \*LPQUERY\_SERVICE\_CONFIGW;
75. typedef struct \_QUERY\_SERVICE\_LOCK\_STATUSW {
76. DWORD fIsLocked;
77. [string,range(0, 8 \* 1024)] LPWSTR lpLockOwner;
78. DWORD dwLockDuration;
79. } QUERY\_SERVICE\_LOCK\_STATUSW,
80. \*LPQUERY\_SERVICE\_LOCK\_STATUSW;
81. typedef struct \_QUERY\_SERVICE\_CONFIGA {
82. DWORD dwServiceType;
83. DWORD dwStartType;
84. DWORD dwErrorControl;
85. [string,range(0, 8 \* 1024)] LPSTR lpBinaryPathName;
86. [string,range(0, 8 \* 1024)] LPSTR lpLoadOrderGroup;
87. DWORD dwTagId;
88. [string,range(0, 8 \* 1024)] LPSTR lpDependencies;
89. [string,range(0, 8 \* 1024)] LPSTR lpServiceStartName;
90. [string,range(0, 8 \* 1024)] LPSTR lpDisplayName;
91. } QUERY\_SERVICE\_CONFIGA,
92. \*LPQUERY\_SERVICE\_CONFIGA;
93. typedef struct {
94. DWORD fIsLocked;
95. [string,range(0, 8 \* 1024)] char\* lpLockOwner;
96. DWORD dwLockDuration;
97. } QUERY\_SERVICE\_LOCK\_STATUSA,
98. \*LPQUERY\_SERVICE\_LOCK\_STATUSA;
99. typedef struct \_SERVICE\_DESCRIPTIONA {
100. [string,range(0, 8 \* 1024)] LPSTR lpDescription;
101. } SERVICE\_DESCRIPTIONA,
102. \*LPSERVICE\_DESCRIPTIONA;
103. typedef [v1\_enum] enum \_SC\_ACTION\_TYPE {
104. SC\_ACTION\_NONE = 0,
105. SC\_ACTION\_RESTART = 1,
106. SC\_ACTION\_REBOOT = 2,
107. SC\_ACTION\_RUN\_COMMAND = 3
108. } SC\_ACTION\_TYPE;
109. typedef struct {
110. SC\_ACTION\_TYPE Type;
111. DWORD Delay;
112. } SC\_ACTION,
113. \*LPSC\_ACTION;
114. typedef struct \_SERVICE\_FAILURE\_ACTIONSA {
115. DWORD dwResetPeriod;
116. [string,range(0, 8 \* 1024)] LPSTR lpRebootMsg;
117. [string,range(0, 8 \* 1024)] LPSTR lpCommand;
118. [range(0, 1024)] DWORD cActions;
119. [size\_is(cActions)] SC\_ACTION \* lpsaActions;
120. } SERVICE\_FAILURE\_ACTIONSA,
121. \*LPSERVICE\_FAILURE\_ACTIONSA;
122. typedef struct \_SERVICE\_DELAYED\_AUTO\_START\_INFO {
123. BOOL fDelayedAutostart;
124. } SERVICE\_DELAYED\_AUTO\_START\_INFO,
125. \*LPSERVICE\_DELAYED\_AUTO\_START\_INFO;
126. typedef struct \_SERVICE\_FAILURE\_ACTIONS\_FLAG {
127. BOOL fFailureActionsOnNonCrashFailures;
128. } SERVICE\_FAILURE\_ACTIONS\_FLAG,
129. \*LPSERVICE\_FAILURE\_ACTIONS\_FLAG;
130. typedef struct \_SERVICE\_SID\_INFO {
131. DWORD dwServiceSidType;
132. } SERVICE\_SID\_INFO,
133. \*LPSERVICE\_SID\_INFO;
134. typedef struct \_SERVICE\_PRESHUTDOWN\_INFO {
135. DWORD dwPreshutdownTimeout;
136. } SERVICE\_PRESHUTDOWN\_INFO,
137. \*LPSERVICE\_PRESHUTDOWN\_INFO;
138. typedef struct \_SERVICE\_DESCRIPTIONW {
139. [string,range(0, 8 \* 1024)] LPWSTR lpDescription;
140. } SERVICE\_DESCRIPTIONW,
141. \*LPSERVICE\_DESCRIPTIONW;
142. typedef struct \_SERVICE\_FAILURE\_ACTIONSW {
143. DWORD dwResetPeriod;
144. [string,range(0, 8 \* 1024)] LPWSTR lpRebootMsg;
145. [string,range(0, 8 \* 1024)] LPWSTR lpCommand;
146. [range(0, 1024)] DWORD cActions;
147. [size\_is(cActions)] SC\_ACTION \* lpsaActions;
148. } SERVICE\_FAILURE\_ACTIONSW,
149. \*LPSERVICE\_FAILURE\_ACTIONSW;
150. typedef [v1\_enum] enum
151. {
152. SC\_STATUS\_PROCESS\_INFO = 0
153. } SC\_STATUS\_TYPE;
154. typedef [v1\_enum] enum
155. {
156. SC\_ENUM\_PROCESS\_INFO = 0
157. } SC\_ENUM\_TYPE;
158. typedef struct \_SERVICE\_PREFERRED\_NODE\_INFO {
159. USHORT usPreferredNode;
160. BOOLEAN fDelete;
161. } SERVICE\_PREFERRED\_NODE\_INFO, \*LPSERVICE\_PREFERRED\_NODE\_INFO;
162. typedef struct \_SERVICE\_TRIGGER\_SPECIFIC\_DATA\_ITEM {
163. DWORD dwDataType;
164. [range(0, 1024)]
165. DWORD cbData;
166. [size\_is(cbData)]
167. PBYTE pData;
168. } SERVICE\_TRIGGER\_SPECIFIC\_DATA\_ITEM, \*PSERVICE\_TRIGGER\_SPECIFIC\_DATA\_ITEM;
169. typedef struct \_SERVICE\_TRIGGER {
170. DWORD dwTriggerType;
171. DWORD dwAction;
172. GUID \* pTriggerSubtype;
173. [range(0, 64)]
174. DWORD cDataItems;
175. [size\_is(cDataItems)]
176. PSERVICE\_TRIGGER\_SPECIFIC\_DATA\_ITEM pDataItems;
177. } SERVICE\_TRIGGER, \*PSERVICE\_TRIGGER;
178. typedef struct \_SERVICE\_TRIGGER\_INFO {
179. [range(0, 64)] DWORD cTriggers;
180. [size\_is(cTriggers)]
181. PSERVICE\_TRIGGER pTriggers;
182. PBYTE pReserved;
183. } SERVICE\_TRIGGER\_INFO, \*PSERVICE\_TRIGGER\_INFO;
184. DWORD
185. RCloseServiceHandle(
186. [in,out] LPSC\_RPC\_HANDLE hSCObject
187. );
188. DWORD
189. RControlService(
190. [in] SC\_RPC\_HANDLE hService,
191. [in] DWORD dwControl,
192. [out] LPSERVICE\_STATUS lpServiceStatus
193. );
194. DWORD
195. RDeleteService(
196. [in] SC\_RPC\_HANDLE hService
197. );
198. DWORD
199. RLockServiceDatabase(
200. [in] SC\_RPC\_HANDLE hSCManager,
201. [out] LPSC\_RPC\_LOCK lpLock
202. );
203. DWORD
204. RQueryServiceObjectSecurity(
205. [in] SC\_RPC\_HANDLE hService,
206. [in] SECURITY\_INFORMATION dwSecurityInformation,
207. [out, size\_is(cbBufSize)] LPBYTE lpSecurityDescriptor,
208. [in, range(0, 1024 \* 256)] DWORD cbBufSize,
209. [out] LPBOUNDED\_DWORD\_256K pcbBytesNeeded
210. );
211. DWORD
212. RSetServiceObjectSecurity(
213. [in] SC\_RPC\_HANDLE hService,
214. [in] SECURITY\_INFORMATION dwSecurityInformation,
215. [in,size\_is(cbBufSize)] LPBYTE lpSecurityDescriptor,
216. [in] DWORD cbBufSize
217. );
218. DWORD
219. RQueryServiceStatus(
220. [in] SC\_RPC\_HANDLE hService,
221. [out] LPSERVICE\_STATUS lpServiceStatus
222. );
223. DWORD
224. RSetServiceStatus(
225. [in] SC\_RPC\_HANDLE hServiceStatus,
226. [in] LPSERVICE\_STATUS lpServiceStatus
227. );
228. DWORD
229. RUnlockServiceDatabase(
230. [in,out] LPSC\_RPC\_LOCK Lock
231. );
232. DWORD
233. RNotifyBootConfigStatus(
234. [in,string,unique,range(0, SC\_MAX\_COMPUTER\_NAME\_LENGTH)]
235. SVCCTL\_HANDLEW lpMachineName,
236. [in] DWORD BootAcceptable
237. );
238. void Opnum10NotUsedOnWire(void);
239. DWORD
240. RChangeServiceConfigW(
241. [in] SC\_RPC\_HANDLE hService,
242. [in] DWORD dwServiceType,
243. [in] DWORD dwStartType,
244. [in] DWORD dwErrorControl,
245. [in,string,unique,range(0, SC\_MAX\_PATH\_LENGTH)]
246. wchar\_t \* lpBinaryPathName,
247. [in,string,unique,range(0, SC\_MAX\_NAME\_LENGTH)]
248. wchar\_t \* lpLoadOrderGroup,
249. [in,out,unique] LPDWORD lpdwTagId,
250. [in,unique,size\_is(dwDependSize)] LPBYTE lpDependencies,
251. [in, range (0, SC\_MAX\_DEPEND\_SIZE)] DWORD dwDependSize,
252. [in,string,unique,range(0, SC\_MAX\_ACCOUNT\_NAME\_LENGTH)]
253. wchar\_t \* lpServiceStartName,
254. [in,unique,size\_is(dwPwSize)] LPBYTE lpPassword,
255. [in, range(0, SC\_MAX\_PWD\_SIZE)] DWORD dwPwSize,
256. [in,string,unique,range(0, SC\_MAX\_NAME\_LENGTH)]
257. wchar\_t \* lpDisplayName
258. );
259. DWORD
260. RCreateServiceW(
261. [in] SC\_RPC\_HANDLE hSCManager,
262. [in,string,range(0, SC\_MAX\_NAME\_LENGTH)]
263. wchar\_t \* lpServiceName,
264. [in,string,unique,range(0, SC\_MAX\_NAME\_LENGTH)]
265. wchar\_t \* lpDisplayName,
266. [in] DWORD dwDesiredAccess,
267. [in] DWORD dwServiceType,
268. [in] DWORD dwStartType,
269. [in] DWORD dwErrorControl,
270. [in,string, range(0, SC\_MAX\_PATH\_LENGTH)]
271. wchar\_t \* lpBinaryPathName,
272. [in,string,unique,range(0, SC\_MAX\_NAME\_LENGTH)]
273. wchar\_t \* lpLoadOrderGroup,
274. [in,out,unique] LPDWORD lpdwTagId,
275. [in,unique,size\_is(dwDependSize)] LPBYTE lpDependencies,
276. [in, range (0, SC\_MAX\_DEPEND\_SIZE)] DWORD dwDependSize,
277. [in,string,unique,range(0, SC\_MAX\_ACCOUNT\_NAME\_LENGTH)]
278. wchar\_t \* lpServiceStartName,
279. [in,unique,size\_is(dwPwSize)] LPBYTE lpPassword,
280. [in, range(0, SC\_MAX\_PWD\_SIZE)] DWORD dwPwSize,
281. [out] LPSC\_RPC\_HANDLE lpServiceHandle
282. );
283. DWORD
284. REnumDependentServicesW(
285. [in] SC\_RPC\_HANDLE hService,
286. [in] DWORD dwServiceState,
287. [out, size\_is(cbBufSize)] LPBYTE lpServices,
288. [in, range(0, 1024 \* 256)] DWORD cbBufSize,
289. [out] LPBOUNDED\_DWORD\_256K pcbBytesNeeded,
290. [out] LPBOUNDED\_DWORD\_256K lpServicesReturned
291. );
292. DWORD
293. REnumServicesStatusW(
294. [in] SC\_RPC\_HANDLE hSCManager,
295. [in] DWORD dwServiceType,
296. [in] DWORD dwServiceState,
297. [out, size\_is(cbBufSize)] LPBYTE lpBuffer,
298. [in, range(0, 1024 \* 256)] DWORD cbBufSize,
299. [out] LPBOUNDED\_DWORD\_256K pcbBytesNeeded,
300. [out] LPBOUNDED\_DWORD\_256K lpServicesReturned,
301. [in,out,unique] LPBOUNDED\_DWORD\_256K lpResumeIndex
302. );
303. DWORD
304. ROpenSCManagerW(
305. [in,string,unique,range(0, SC\_MAX\_COMPUTER\_NAME\_LENGTH)]
306. SVCCTL\_HANDLEW lpMachineName,
307. [in,string,unique,range(0, SC\_MAX\_NAME\_LENGTH)]
308. wchar\_t \* lpDatabaseName,
309. [in] DWORD dwDesiredAccess,
310. [out] LPSC\_RPC\_HANDLE lpScHandle
311. );
312. DWORD
313. ROpenServiceW(
314. [in] SC\_RPC\_HANDLE hSCManager,
315. [in,string,range(0, SC\_MAX\_NAME\_LENGTH)]
316. wchar\_t \* lpServiceName,
317. [in] DWORD dwDesiredAccess,
318. [out] LPSC\_RPC\_HANDLE lpServiceHandle
319. );
320. DWORD
321. RQueryServiceConfigW(
322. [in] SC\_RPC\_HANDLE hService,
323. [out] LPQUERY\_SERVICE\_CONFIGW lpServiceConfig,
324. [in, range(0, 1024 \* 8)] DWORD cbBufSize,
325. [out] LPBOUNDED\_DWORD\_8K pcbBytesNeeded
326. );
327. DWORD
328. RQueryServiceLockStatusW(
329. [in] SC\_RPC\_HANDLE hSCManager,
330. [out] LPQUERY\_SERVICE\_LOCK\_STATUSW lpLockStatus,
331. [in, range(0, 1024 \* 4)] DWORD cbBufSize,
332. [out] LPBOUNDED\_DWORD\_4K pcbBytesNeeded
333. );
334. DWORD
335. RStartServiceW(
336. [in] SC\_RPC\_HANDLE hService,
337. [in, range(0, SC\_MAX\_ARGUMENTS)] DWORD argc,
338. [in,unique,size\_is(argc)] LPSTRING\_PTRSW argv
339. );
340. DWORD
341. RGetServiceDisplayNameW(
342. [in] SC\_RPC\_HANDLE hSCManager,
343. [in,string,range(0, SC\_MAX\_NAME\_LENGTH)]
344. wchar\_t \* lpServiceName,
345. [out,string, range(1, 4\*1024+1), size\_is(\*lpcchBuffer+1)]
346. wchar\_t \* lpDisplayName,
347. [in,out] DWORD \* lpcchBuffer
348. );
349. DWORD
350. RGetServiceKeyNameW(
351. [in] SC\_RPC\_HANDLE hSCManager,
352. [in,string,range(0, SC\_MAX\_NAME\_LENGTH)]
353. wchar\_t \* lpDisplayName,
354. [out,string, range(1, 4\*1024+1), size\_is(\*lpcchBuffer+1)]
355. wchar\_t \* lpServiceName,
356. [in,out] DWORD \* lpcchBuffer
357. );
358. void Opnum22NotUsedOnWire(void);
359. DWORD
360. RChangeServiceConfigA(
361. [in] SC\_RPC\_HANDLE hService,
362. [in] DWORD dwServiceType,
363. [in] DWORD dwStartType,
364. [in] DWORD dwErrorControl,
365. [in,string,unique,range(0, SC\_MAX\_PATH\_LENGTH)]
366. LPSTR lpBinaryPathName,
367. [in,string,unique,range(0, SC\_MAX\_NAME\_LENGTH)]
368. LPSTR lpLoadOrderGroup,
369. [in,out,unique] LPDWORD lpdwTagId,
370. [in,unique,size\_is(dwDependSize)] LPBYTE lpDependencies,
371. [in, range (0, SC\_MAX\_DEPEND\_SIZE)] DWORD dwDependSize,
372. [in,string,unique,range(0, SC\_MAX\_ACCOUNT\_NAME\_LENGTH)]
373. LPSTR lpServiceStartName,
374. [in,unique,size\_is(dwPwSize)] LPBYTE lpPassword,
375. [in, range(0, SC\_MAX\_PWD\_SIZE)] DWORD dwPwSize,
376. [in,string,unique,range(0, SC\_MAX\_NAME\_LENGTH)]
377. LPSTR lpDisplayName
378. );
379. DWORD
380. RCreateServiceA(
381. [in] SC\_RPC\_HANDLE hSCManager,
382. [in,string,range(0, SC\_MAX\_NAME\_LENGTH)]
383. LPSTR lpServiceName,
384. [in,string,unique,range(0, SC\_MAX\_NAME\_LENGTH)]
385. LPSTR lpDisplayName,
386. [in] DWORD dwDesiredAccess,
387. [in] DWORD dwServiceType,
388. [in] DWORD dwStartType,
389. [in] DWORD dwErrorControl,
390. [in,string, range(0, SC\_MAX\_PATH\_LENGTH)]
391. LPSTR lpBinaryPathName,
392. [in,string,unique,range(0, SC\_MAX\_NAME\_LENGTH)]
393. LPSTR lpLoadOrderGroup,
394. [in,out,unique] LPDWORD lpdwTagId,
395. [in,unique,size\_is(dwDependSize)] LPBYTE lpDependencies,
396. [in, range (0, SC\_MAX\_DEPEND\_SIZE)] DWORD dwDependSize,
397. [in,string,unique,range(0, SC\_MAX\_ACCOUNT\_NAME\_LENGTH)]
398. LPSTR lpServiceStartName,
399. [in,unique,size\_is(dwPwSize)] LPBYTE lpPassword,
400. [in, range(0, SC\_MAX\_PWD\_SIZE)] DWORD dwPwSize,
401. [out] LPSC\_RPC\_HANDLE lpServiceHandle
402. );
403. DWORD
404. REnumDependentServicesA(
405. [in] SC\_RPC\_HANDLE hService,
406. [in] DWORD dwServiceState,
407. [out, size\_is(cbBufSize)] LPBYTE lpServices,
408. [in, range(0, 1024 \* 256)] DWORD cbBufSize,
409. [out] LPBOUNDED\_DWORD\_256K pcbBytesNeeded,
410. [out] LPBOUNDED\_DWORD\_256K lpServicesReturned
411. );
412. DWORD
413. REnumServicesStatusA(
414. [in] SC\_RPC\_HANDLE hSCManager,
415. [in] DWORD dwServiceType,
416. [in] DWORD dwServiceState,
417. [out, size\_is(cbBufSize)] LPBYTE lpBuffer,
418. [in, range(0, 1024 \* 256)] DWORD cbBufSize,
419. [out] LPBOUNDED\_DWORD\_256K pcbBytesNeeded,
420. [out] LPBOUNDED\_DWORD\_256K lpServicesReturned,
421. [in,out,unique] LPBOUNDED\_DWORD\_256K lpResumeIndex
422. );
423. DWORD
424. ROpenSCManagerA(
425. [in,string,unique,range(0, SC\_MAX\_COMPUTER\_NAME\_LENGTH)]
426. SVCCTL\_HANDLEA lpMachineName,
427. [in,string,unique,range(0, SC\_MAX\_NAME\_LENGTH)]
428. LPSTR lpDatabaseName,
429. [in] DWORD dwDesiredAccess,
430. [out] LPSC\_RPC\_HANDLE lpScHandle
431. );
432. DWORD
433. ROpenServiceA(
434. [in] SC\_RPC\_HANDLE hSCManager,
435. [in,string,range(0, SC\_MAX\_NAME\_LENGTH)]
436. LPSTR lpServiceName,
437. [in] DWORD dwDesiredAccess,
438. [out] LPSC\_RPC\_HANDLE lpServiceHandle
439. );
440. DWORD
441. RQueryServiceConfigA(
442. [in] SC\_RPC\_HANDLE hService,
443. [out] LPQUERY\_SERVICE\_CONFIGA lpServiceConfig,
444. [in, range(0, 1024 \* 8)] DWORD cbBufSize,
445. [out] LPBOUNDED\_DWORD\_8K pcbBytesNeeded
446. );
447. DWORD
448. RQueryServiceLockStatusA(
449. [in] SC\_RPC\_HANDLE hSCManager,
450. [out] LPQUERY\_SERVICE\_LOCK\_STATUSA lpLockStatus,
451. [in, range(0, 1024 \* 4)] DWORD cbBufSize,
452. [out] LPBOUNDED\_DWORD\_4K pcbBytesNeeded
453. );
454. DWORD
455. RStartServiceA(
456. [in] SC\_RPC\_HANDLE hService,
457. [in, range(0, SC\_MAX\_ARGUMENTS)] DWORD argc,
458. [in,unique,size\_is(argc)] LPSTRING\_PTRSA argv
459. );
460. DWORD
461. RGetServiceDisplayNameA(
462. [in] SC\_RPC\_HANDLE hSCManager,
463. [in,string,range(0, SC\_MAX\_NAME\_LENGTH)] LPSTR lpServiceName,
464. [out,string, size\_is(\*lpcchBuffer)] LPSTR lpDisplayName,
465. [in,out] LPBOUNDED\_DWORD\_4K lpcchBuffer
466. );
467. DWORD
468. RGetServiceKeyNameA(
469. [in] SC\_RPC\_HANDLE hSCManager,
470. [in,string,range(0, SC\_MAX\_NAME\_LENGTH)] LPSTR lpDisplayName,
471. [out,string, size\_is(\*lpcchBuffer)] LPSTR lpKeyName,
472. [in,out] LPBOUNDED\_DWORD\_4K lpcchBuffer
473. );
474. void Opnum34NotUsedOnWire(void);
475. DWORD
476. REnumServiceGroupW(
477. [in] SC\_RPC\_HANDLE hSCManager,
478. [in] DWORD dwServiceType,
479. [in] DWORD dwServiceState,
480. [out, size\_is(cbBufSize)] LPBYTE lpBuffer,
481. [in, range(0, 1024 \* 256)] DWORD cbBufSize,
482. [out] LPBOUNDED\_DWORD\_256K pcbBytesNeeded,
483. [out] LPBOUNDED\_DWORD\_256K lpServicesReturned,
484. [in,out,unique] LPBOUNDED\_DWORD\_256K lpResumeIndex,
485. [in,string,unique,range(0, SC\_MAX\_NAME\_LENGTH)]
486. LPCWSTR pszGroupName
487. );
488. typedef struct \_SERVICE\_RPC\_REQUIRED\_PRIVILEGES\_INFO
489. {
490. [range(0, 1024 \* 4)] DWORD cbRequiredPrivileges;
491. [size\_is(cbRequiredPrivileges)] PBYTE pRequiredPrivileges;
492. } SERVICE\_RPC\_REQUIRED\_PRIVILEGES\_INFO,
493. \*LPSERVICE\_RPC\_REQUIRED\_PRIVILEGES\_INFO;
494. typedef struct \_SC\_RPC\_CONFIG\_INFOA
495. {
496. DWORD dwInfoLevel;
497. [switch\_is(dwInfoLevel)] union
498. {
499. [case(1)]
500. LPSERVICE\_DESCRIPTIONA psd;
501. [case(2)]
502. LPSERVICE\_FAILURE\_ACTIONSA psfa;
503. [case(3)]
504. LPSERVICE\_DELAYED\_AUTO\_START\_INFO psda;
505. [case(4)]
506. LPSERVICE\_FAILURE\_ACTIONS\_FLAG psfaf;
507. [case(5)]
508. LPSERVICE\_SID\_INFO pssid;
509. [case(6)]
510. LPSERVICE\_RPC\_REQUIRED\_PRIVILEGES\_INFO psrp;
511. [case(7)]
512. LPSERVICE\_PRESHUTDOWN\_INFO psps;
513. [case(8)]
514. PSERVICE\_TRIGGER\_INFO psti;
515. [case(9)]
516. LPSERVICE\_PREFERRED\_NODE\_INFO pspn;
517. };
518. } SC\_RPC\_CONFIG\_INFOA;
519. typedef struct \_SC\_RPC\_CONFIG\_INFOW
520. {
521. DWORD dwInfoLevel;
522. [switch\_is(dwInfoLevel)] union
523. {
524. [case(1)]
525. LPSERVICE\_DESCRIPTIONW psd;
526. [case(2)]
527. LPSERVICE\_FAILURE\_ACTIONSW psfa;
528. [case(3)]
529. LPSERVICE\_DELAYED\_AUTO\_START\_INFO psda;
530. [case(4)]
531. LPSERVICE\_FAILURE\_ACTIONS\_FLAG psfaf;
532. [case(5)]
533. LPSERVICE\_SID\_INFO pssid;
534. [case(6)]
535. LPSERVICE\_RPC\_REQUIRED\_PRIVILEGES\_INFO psrp;
536. [case(7)]
537. LPSERVICE\_PRESHUTDOWN\_INFO psps;
538. [case(8)]
539. PSERVICE\_TRIGGER\_INFO psti;
540. [case(9)]
541. LPSERVICE\_PREFERRED\_NODE\_INFO pspn;
542. };
543. } SC\_RPC\_CONFIG\_INFOW;
544. DWORD
545. RChangeServiceConfig2A(
546. [in] SC\_RPC\_HANDLE hService,
547. [in] SC\_RPC\_CONFIG\_INFOA Info
548. );
549. DWORD
550. RChangeServiceConfig2W(
551. [in] SC\_RPC\_HANDLE hService,
552. [in] SC\_RPC\_CONFIG\_INFOW Info
553. );
554. DWORD
555. RQueryServiceConfig2A(
556. [in] SC\_RPC\_HANDLE hService,
557. [in] DWORD dwInfoLevel,
558. [out, size\_is(cbBufSize)] LPBYTE lpBuffer,
559. [in, range(0, 1024 \* 8)] DWORD cbBufSize,
560. [out] LPBOUNDED\_DWORD\_8K pcbBytesNeeded
561. );
562. DWORD
563. RQueryServiceConfig2W(
564. [in] SC\_RPC\_HANDLE hService,
565. [in] DWORD dwInfoLevel,
566. [out, size\_is(cbBufSize)] LPBYTE lpBuffer,
567. [in, range(0, 1024 \* 8)] DWORD cbBufSize,
568. [out] LPBOUNDED\_DWORD\_8K pcbBytesNeeded
569. );
570. DWORD
571. RQueryServiceStatusEx(
572. [in] SC\_RPC\_HANDLE hService,
573. [in] SC\_STATUS\_TYPE InfoLevel,
574. [out, size\_is(cbBufSize)] LPBYTE lpBuffer,
575. [in, range(0, 1024 \* 8)] DWORD cbBufSize,
576. [out] LPBOUNDED\_DWORD\_8K pcbBytesNeeded
577. );
578. DWORD
579. REnumServicesStatusExA (
580. [in] SC\_RPC\_HANDLE hSCManager,
581. [in] SC\_ENUM\_TYPE InfoLevel,
582. [in] DWORD dwServiceType,
583. [in] DWORD dwServiceState,
584. [out, size\_is(cbBufSize)] LPBYTE lpBuffer,
585. [in, range(0, 1024 \* 256)] DWORD cbBufSize,
586. [out] LPBOUNDED\_DWORD\_256K pcbBytesNeeded,
587. [out] LPBOUNDED\_DWORD\_256K lpServicesReturned,
588. [in,out,unique] LPBOUNDED\_DWORD\_256K lpResumeIndex,
589. [in,string,unique,range(0, SC\_MAX\_NAME\_LENGTH)]
590. LPCSTR pszGroupName
591. );
592. DWORD
593. REnumServicesStatusExW (
594. [in] SC\_RPC\_HANDLE hSCManager,
595. [in] SC\_ENUM\_TYPE InfoLevel,
596. [in] DWORD dwServiceType,
597. [in] DWORD dwServiceState,
598. [out, size\_is(cbBufSize)] LPBYTE lpBuffer,
599. [in, range(0, 1024 \* 256)] DWORD cbBufSize,
600. [out] LPBOUNDED\_DWORD\_256K pcbBytesNeeded,
601. [out] LPBOUNDED\_DWORD\_256K lpServicesReturned,
602. [in,out,unique] LPBOUNDED\_DWORD\_256K lpResumeIndex,
603. [in,string,unique,range(0, SC\_MAX\_NAME\_LENGTH)]
604. LPCWSTR pszGroupName
605. );
606. void Opnum43NotUsedOnWire(void);
607. DWORD
608. RCreateServiceWOW64A(
609. [in] SC\_RPC\_HANDLE hSCManager,
610. [in,string,range(0, SC\_MAX\_NAME\_LENGTH)]
611. LPSTR lpServiceName,
612. [in,string,unique,range(0, SC\_MAX\_NAME\_LENGTH)]
613. LPSTR lpDisplayName,
614. [in] DWORD dwDesiredAccess,
615. [in] DWORD dwServiceType,
616. [in] DWORD dwStartType,
617. [in] DWORD dwErrorControl,
618. [in,string, range(0, SC\_MAX\_PATH\_LENGTH)]
619. LPSTR lpBinaryPathName,
620. [in,string,unique,range(0, SC\_MAX\_NAME\_LENGTH)]
621. LPSTR lpLoadOrderGroup,
622. [in,out,unique] LPDWORD lpdwTagId,
623. [in,unique,size\_is(dwDependSize)] LPBYTE lpDependencies,
624. [in, range (0, SC\_MAX\_DEPEND\_SIZE)] DWORD dwDependSize,
625. [in,string,unique,range(0, SC\_MAX\_ACCOUNT\_NAME\_LENGTH)]
626. LPSTR lpServiceStartName,
627. [in,unique,size\_is(dwPwSize)] LPBYTE lpPassword,
628. [in, range(0, SC\_MAX\_PWD\_SIZE)] DWORD dwPwSize,
629. [out] LPSC\_RPC\_HANDLE lpServiceHandle
630. );
631. DWORD
632. RCreateServiceWOW64W(
633. [in] SC\_RPC\_HANDLE hSCManager,
634. [in,string,range(0, SC\_MAX\_NAME\_LENGTH)]
635. wchar\_t \* lpServiceName,
636. [in,string,unique,range(0, SC\_MAX\_NAME\_LENGTH)]
637. wchar\_t \* lpDisplayName,
638. [in] DWORD dwDesiredAccess,
639. [in] DWORD dwServiceType,
640. [in] DWORD dwStartType,
641. [in] DWORD dwErrorControl,
642. [in,string,range(0, SC\_MAX\_PATH\_LENGTH)]
643. wchar\_t \* lpBinaryPathName,
644. [in,string,unique,range(0, SC\_MAX\_NAME\_LENGTH)]
645. wchar\_t \* lpLoadOrderGroup,
646. [in,out,unique] LPDWORD lpdwTagId,
647. [in,unique,size\_is(dwDependSize)] LPBYTE lpDependencies,
648. [in, range (0, SC\_MAX\_DEPEND\_SIZE)] DWORD dwDependSize,
649. [in,string,unique,range(0, SC\_MAX\_ACCOUNT\_NAME\_LENGTH)]
650. wchar\_t \* lpServiceStartName,
651. [in,unique,size\_is(dwPwSize)] LPBYTE lpPassword,
652. [in, range(0, SC\_MAX\_PWD\_SIZE)] DWORD dwPwSize,
653. [out] LPSC\_RPC\_HANDLE lpServiceHandle
654. );
655. void Opnum46NotUsedOnWire(void);
657. typedef struct \_SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_1
658. {
659. ULONGLONG ullThreadId;
660. DWORD dwNotifyMask;
661. UCHAR CallbackAddressArray [ 16 ];
662. UCHAR CallbackParamAddressArray [ 16 ];
663. SERVICE\_STATUS\_PROCESS ServiceStatus;
664. DWORD dwNotificationStatus;
665. DWORD dwSequence;
666. } SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_1,
667. \*PSERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_1;
668. typedef struct \_SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_2
669. {
670. ULONGLONG ullThreadId;
671. DWORD dwNotifyMask;
672. UCHAR CallbackAddressArray [ 16 ];
673. UCHAR CallbackParamAddressArray [ 16 ];
674. SERVICE\_STATUS\_PROCESS ServiceStatus;
675. DWORD dwNotificationStatus;
676. DWORD dwSequence;
677. DWORD dwNotificationTriggered;
678. [string, range(0, 64\*1024)] PWSTR pszServiceNames;
679. } SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_2,
680. \*PSERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_2;
681. typedef SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_2
682. SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS,
683. \*PSERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS;
684. typedef struct \_SC\_RPC\_NOTIFY\_PARAMS
685. {
686. DWORD dwInfoLevel;
687. [ switch\_is ( dwInfoLevel ) ]
688. union
689. {
690. [case(1)]
691. PSERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_1 pStatusChangeParam1;
692. [case(2)]
693. PSERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_2 pStatusChangeParams;
694. };
695. } SC\_RPC\_NOTIFY\_PARAMS;
696. typedef struct \_SC\_RPC\_NOTIFY\_PARAMS\_LIST
697. {
698. BOUNDED\_DWORD\_4K cElements;
699. [size\_is(cElements)] SC\_RPC\_NOTIFY\_PARAMS NotifyParamsArray [\*];
700. } SC\_RPC\_NOTIFY\_PARAMS\_LIST, \*PSC\_RPC\_NOTIFY\_PARAMS\_LIST;
701. DWORD
702. RNotifyServiceStatusChange(
703. [in] SC\_RPC\_HANDLE hService,
704. [in] SC\_RPC\_NOTIFY\_PARAMS NotifyParams,
705. [in] GUID \* pClientProcessGuid,
706. [out] GUID \* pSCMProcessGuid,
707. [out] PBOOL pfCreateRemoteQueue,
708. [out] LPSC\_NOTIFY\_RPC\_HANDLE phNotify
709. );
710. error\_status\_t
711. RGetNotifyResults(
712. [in] SC\_NOTIFY\_RPC\_HANDLE hNotify,
713. [out] PSC\_RPC\_NOTIFY\_PARAMS\_LIST \*ppNotifyParams
714. );
715. DWORD
716. RCloseNotifyHandle(
717. [in, out] LPSC\_NOTIFY\_RPC\_HANDLE phNotify,
718. [out] PBOOL pfApcFired
719. );
720. typedef struct \_SERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSA
721. {
722. DWORD dwReason;
723. [string,range(0, SC\_MAX\_COMMENT\_LENGTH)] LPSTR pszComment;
724. } SERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSA,
725. \*PSERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSA;
726. typedef struct \_SERVICE\_CONTROL\_STATUS\_REASON\_OUT\_PARAMS
727. {
728. SERVICE\_STATUS\_PROCESS ServiceStatus;
729. } SERVICE\_CONTROL\_STATUS\_REASON\_OUT\_PARAMS,
730. \*PSERVICE\_CONTROL\_STATUS\_REASON\_OUT\_PARAMS;
731. typedef [switch\_type(DWORD)]
732. union \_SC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSA
733. {
734. [case(1)]
735. PSERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSA psrInParams;
736. } SC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSA,
737. \*PSC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSA;
738. typedef [switch\_type(DWORD)]
739. union \_SC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSA
740. {
741. [case(1)]
742. PSERVICE\_CONTROL\_STATUS\_REASON\_OUT\_PARAMS psrOutParams;
743. } SC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSA,
744. \*PSC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSA;
745. DWORD
746. RControlServiceExA (
747. [in] SC\_RPC\_HANDLE hService,
748. [in] DWORD dwControl,
749. [in] DWORD dwInfoLevel,
750. [in, switch\_is(dwInfoLevel)]
751. PSC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSA pControlInParams,
752. [out, switch\_is(dwInfoLevel)]
753. PSC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSA pControlOutParams
754. );
755. typedef struct \_SERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSW
756. {
757. DWORD dwReason;
758. [string,range(0, SC\_MAX\_COMMENT\_LENGTH)] LPWSTR pszComment;
759. } SERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSW,
760. \*PSERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSW;
761. typedef [switch\_type(DWORD)]
762. union \_SC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSW
763. {
764. [case(1)]
765. PSERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSW psrInParams;
766. } SC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSW,
767. \*PSC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSW;
768. typedef [switch\_type(DWORD)]
769. union \_SC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSW
770. {
771. [case(1)]
772. PSERVICE\_CONTROL\_STATUS\_REASON\_OUT\_PARAMS psrOutParams;
773. } SC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSW,
774. \*PSC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSW;
775. DWORD
776. RControlServiceExW (
777. [in] SC\_RPC\_HANDLE hService,
778. [in] DWORD dwControl,
779. [in] DWORD dwInfoLevel,
780. [in, switch\_is(dwInfoLevel)]
781. PSC\_RPC\_SERVICE\_CONTROL\_IN\_PARAMSW pControlInParams,
782. [out, switch\_is(dwInfoLevel)]
783. PSC\_RPC\_SERVICE\_CONTROL\_OUT\_PARAMSW pControlOutParams
784. );
785. void Opnum52NotUsedOnWire(void);
786. void Opnum53NotUsedOnWire(void);
787. void Opnum54NotUsedOnWire(void);
788. void Opnum55NotUsedOnWire(void);
789. DWORD
790. RQueryServiceConfigEx (
791. [in] SC\_RPC\_HANDLE hService,
792. [in] DWORD dwInfoLevel,
793. [out] SC\_RPC\_CONFIG\_INFOW \* pInfo
794. );
795. }

# Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs.

The terms "earlier" and "later", when used with a product version, refer to either all preceding versions or all subsequent versions, respectively. The term "through" refers to the inclusive range of versions. Applicable Microsoft products are listed chronologically in this section.

**Windows Client**

* Windows NT operating system
* Windows 2000 Professional operating system
* Windows XP operating system
* Windows Vista operating system
* Windows 7 operating system
* Windows 8 operating system
* Windows 8.1 operating system
* Windows 10 operating system

**Windows Server**

* Windows 2000 Server operating system
* Windows Server 2003 operating system
* Windows Server 2003 R2 operating system
* Windows Server 2008 operating system
* Windows Server 2008 R2 operating system
* Windows Server 2012 operating system
* Windows Server 2012 R2 operating system
* Windows Server 2016 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms "SHOULD" or "SHOULD NOT" implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term "MAY" implies that the product does not follow the prescription.

[<1> Section 2.1.2](#Appendix_A_Target_1): An authentication level of RPC\_C\_AUTHN\_LEVEL\_PKT\_PRIVACY is used only in Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2 operating system.

[<2> Section 2.2.21](#Appendix_A_Target_2): The structures are not available in Windows NT, Windows 2000 operating system, Windows XP, and Windows Server 2003.

[<3> Section 2.2.22](#Appendix_A_Target_3): RPC\_CONFIG\_INFOW is not available in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<4> Section 2.2.22](#Appendix_A_Target_4): psti is not available in Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2003 R2, or Windows Server 2008.

[<5> Section 2.2.22](#Appendix_A_Target_5): pspn is not available in Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2003 R2, or Windows Server 2008.

[<6> Section 2.2.23](#Appendix_A_Target_6): The SC\_RPC\_NOTIFY\_PARAMS structure is not available in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<7> Section 2.2.24](#Appendix_A_Target_7): The [SC\_RPC\_NOTIFY\_PARAMS\_LIST](#Section_46063c6407cb4055985c75fffb259f41) structure is not available in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<8> Section 2.2.30](#Appendix_A_Target_8): The SERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSA structure is not available in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<9> Section 2.2.31](#Appendix_A_Target_9): The [SERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSW](#Section_1ac2684aea98420cac337441af91282e) structure is not available in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<10> Section 2.2.32](#Appendix_A_Target_10): The [SERVICE\_CONTROL\_STATUS\_REASON\_OUT\_PARAMS](#Section_2679fdcc5e6a40928958a625f2a4ace7) structure is not available in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<11> Section 2.2.33](#Appendix_A_Target_11): The [SERVICE\_DELAYED\_AUTO\_START\_INFO](#Section_805b8296863d4d1e8ae8f639adf8c6cb) structure is not available in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<12> Section 2.2.41](#Appendix_A_Target_12): The [SERVICE\_FAILURE\_ACTIONS\_FLAG](#Section_9b244e2e82fc4c548f4fb19034faa2c4) structure is not available in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<13> Section 2.2.42](#Appendix_A_Target_13): Not available in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<14> Section 2.2.44](#Appendix_A_Target_14): Not available in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<15> Section 2.2.45](#Appendix_A_Target_15): Not available in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<16> Section 2.2.46](#Appendix_A_Target_16): Not available in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<17> Section 2.2.47](#Appendix_A_Target_17): Not available in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<18> Section 2.2.47](#Appendix_A_Target_18): Not available in Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, or Windows Server 2003 R2, or Windows Server 2008.

[<19> Section 2.2.47](#Appendix_A_Target_19): Not available in Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, or Windows Server 2003 R2, or Windows Server 2008.

[<20> Section 2.2.47](#Appendix_A_Target_20): Windows services indicate service-specific error codes by setting **dwWin32ExitCode** to ERROR\_SERVICE\_SPECIFIC\_ERROR (1066) and setting the specific error in the **dwServiceSpecificExitCode** member.

[<21> Section 2.2.48](#Appendix_A_Target_21): Not available in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<22> Section 2.2.49](#Appendix_A_Target_22): Not available in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<23> Section 2.2.49](#Appendix_A_Target_23): Available in Windows 7 and Windows Server 2008 R2.

[<24> Section 2.2.49](#Appendix_A_Target_24): Available in Windows 7 and Windows Server 2008 R2.

[<25> Section 2.2.52](#Appendix_A_Target_25): Available in Windows 7 and Windows Server 2008 R2.

[<26> Section 2.2.53](#Appendix_A_Target_26): Available in Windows 7 and Windows Server 2008 R2.

[<27> Section 2.2.54](#Appendix_A_Target_27): Available in Windows 7 and Windows Server 2008 R2.

[<28> Section 2.2.55](#Appendix_A_Target_28): Available in Windows 7 and Windows Server 2008 R2.

[<29> Section 3.1.1](#Appendix_A_Target_29): In Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2008 R2, localized strings are not supported.

[<30> Section 3.1.1](#Appendix_A_Target_30): Available in Windows 7 and Windows Server 2008 R2.

[<31> Section 3.1.1](#Appendix_A_Target_31): Available in Windows 7 and Windows Server 2008 R2.

[<32> Section 3.1.4](#Appendix_A_Target_32): Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2 clients use multiplexed [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) connections for [RGetNotifyResults](#Section_7020f3e117b9495680dc583ea1509fe6) on request if the server supports them, and they fall back to non-multiplexed connections if the server doesn't support multiplexed connections.

[<33> Section 3.1.4](#Appendix_A_Target_33): Available in Windows 7 and Windows Server 2008 R2.

[<34> Section 3.1.4](#Appendix_A_Target_34): Gaps in the [**opnum**](#gt_e127848e-c66d-427d-b3aa-9f904fa4ada7) numbering sequence apply to Windows as follows.

| Opnum | Description |
| --- | --- |
| 10 | Only used locally by Windows, never remotely. |
| 22 | Only used locally by Windows, never remotely. |
| 34 | Only used locally by Windows, never remotely. |
| 43 | Only used locally by Windows, never remotely. |
| 46 | Only used locally by Windows, never remotely. |
| 52 | Only used locally by Windows, never remotely. |
| 53 | Only used locally by Windows, never remotely. |
| 54 | Only used locally by Windows, never remotely. |
| 55 | Only used locally by Windows, never remotely. |

[<35> Section 3.1.4.2](#Appendix_A_Target_35): Windows waits 30 seconds for the service to respond.

[<36> Section 3.1.4.4](#Appendix_A_Target_36): In Windows NT, Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2, after the database is locked, the server does not allow further client operations on the database until it is unlocked. In Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, the server ignores the database lock.

In Windows NT 3.51 operating system, Windows NT 4.0 operating system, Windows 2000, Windows 2000 Server, Windows Server 2003, Windows Server 2003 R2, and Windows XP, the server responds with the error code ERROR\_SERVICE\_DATABASE\_LOCKED (1055) for [RStartServiceA (section 3.1.4.30)](#Section_f15fc39185764e30b15875c306e1cba2) and [RStartServiceW (section 3.1.4.19)](#Section_d9be95a2cf014bdcb30f6fe4b37ada16) RPCs if the database has been locked using [RLockServiceDatabase (section 3.1.4.4)](#Section_ff71f732e91d41898fb9a410674c63ad).

In Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, the server does not respond with error code ERROR\_SERVICE\_DATABASE\_LOCKED (1055) for RStartServiceA (section 3.1.4.30) and RStartServiceW (section 3.1.4.19) RPCs after the database is locked using RLockServiceDatabase (section 3.1.4.4).

[<37> Section 3.1.4.4](#Appendix_A_Target_37): In Windows NT, Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2, after the database is locked, the server does not allow further client operations on the database until it is unlocked. In Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, the server ignores the database lock.

In Windows NT 3.51, Windows NT 4.0, Windows 2000, Windows 2000 Server, Windows Server 2003, Windows Server 2003 R2, and Windows XP, the server responds with the error code ERROR\_SERVICE\_DATABASE\_LOCKED (1055) for RStartServiceA (section 3.1.4.30) and RStartServiceW (section 3.1.4.19) RPCs if the database has been locked using RLockServiceDatabase (section 3.1.4.4).

In Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, the server does not respond with error code ERROR\_SERVICE\_DATABASE\_LOCKED (1055) for RStartServiceA (section 3.1.4.30) and RStartServiceW (section 3.1.4.19) RPCs after the database is locked using RLockServiceDatabase (section 3.1.4.4).

[<38> Section 3.1.4.9](#Appendix_A_Target_38): In Windows NT, Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2, after the database is locked, the server does not allow further client operations on the database until it is unlocked. In Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, the server ignores the database lock.

In Windows NT 3.51, Windows NT 4.0, Windows 2000, Windows 2000 Server, Windows Server 2003, Windows Server 2003 R2, and Windows XP, the server responds with the error code ERROR\_SERVICE\_DATABASE\_LOCKED (1055) for RStartServiceA (section 3.1.4.30) and RStartServiceW (section 3.1.4.19) RPCs if the database has been locked using RLockServiceDatabase (section 3.1.4.4).

In Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, the server does not respond with error code ERROR\_SERVICE\_DATABASE\_LOCKED (1055) for RStartServiceA (section 3.1.4.30) and RStartServiceW (section 3.1.4.19) RPCs after the database is locked using RLockServiceDatabase (section 3.1.4.4).

[<39> Section 3.1.4.11](#Appendix_A_Target_39): Windows fails the request with ERROR\_INVALID\_PARAMETER (87) if the client tries to change the *dwServiceType* to SERVICE\_FILE\_SYSTEM\_DRIVER or SERVICE\_KERNEL\_DRIVER.

[<40> Section 3.1.4.15](#Appendix_A_Target_40): Windows fails the request with ERROR\_ACCESS\_DENIED (5) if the client does not have sufficient access rights or for operations that do not match the granted access right.

[<41> Section 3.1.4.19](#Appendix_A_Target_41): In Windows NT, Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2, after the database is locked, the server does not allow further client operations on the database until it is unlocked. In Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, the server ignores the database lock.

In Windows NT 3.51, Windows NT 4.0, Windows 2000, Windows 2000 Server, Windows XP, Windows Server 2003, and Windows Server 2003 R2, the server responds with the error code ERROR\_SERVICE\_DATABASE\_LOCKED (1055) for RStartServiceA (section 3.1.4.30) and RStartServiceW (section 3.1.4.19) RPCs if the database has been locked using RLockServiceDatabase (section 3.1.4.4).

In Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, the server does not respond with error code ERROR\_SERVICE\_DATABASE\_LOCKED (1055) for RStartServiceA (section 3.1.4.30) and RStartServiceW (section 3.1.4.19) RPCs after the database is locked using RLockServiceDatabase (section 3.1.4.4).

[<42> Section 3.1.4.19](#Appendix_A_Target_42): Windows waits 30 seconds for the service to respond.

[<43> Section 3.1.4.19](#Appendix_A_Target_43): In Windows NT, Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2, after the database is locked, the server does not allow further client operations on the database until it is unlocked. In Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, the server ignores the database lock.

In Windows NT 3.51, Windows NT 4.0, Windows 2000, Windows 2000 Server, Windows XP, Windows Server 2003, and Windows Server 2003 R2, the server responds with the error code ERROR\_SERVICE\_DATABASE\_LOCKED (1055) for RStartServiceA (section 3.1.4.30) and RStartServiceW (section 3.1.4.19) RPCs if the database has been locked using RLockServiceDatabase (section 3.1.4.4).

In Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, the server does not respond with error code ERROR\_SERVICE\_DATABASE\_LOCKED (1055) for RStartServiceA (section 3.1.4.30) and RStartServiceW (section 3.1.4.19) RPCs after the database is locked using RLockServiceDatabase (section 3.1.4.4).

[<44> Section 3.1.4.22](#Appendix_A_Target_44): Windows fails the request with ERROR\_INVALID\_PARAMETER (87) if the client tries to change *dwServiceType* to SERVICE\_FILE\_SYSTEM\_DRIVER or SERVICE\_KERNEL\_DRIVER.

[<45> Section 3.1.4.26](#Appendix_A_Target_45): Windows fails the request with ERROR\_ACCESS\_DENIED (5) if the client does not have sufficient access rights or for operations that do not match the granted access right.

[<46> Section 3.1.4.30](#Appendix_A_Target_46): In Windows NT, Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2, after the database is locked, the server does not allow further client operations on the database until it is unlocked. In Windows Vista and later and Windows Server 2008 and later, the server ignores the database lock.

In Windows NT 3.51, Windows NT 4.0, Windows 2000, Windows 2000 Server, Windows XP, Windows Server 2003, and Windows Server 2003 R2, the server responds with error code ERROR\_SERVICE\_DATABASE\_LOCKED (1055) for RStartServiceA (section 3.1.4.30) and RStartServiceW (section 3.1.4.19) RPCs if the database has been locked using RLockServiceDatabase (section 3.1.4.4).

In Windows Vista and later and Windows Server 2008 and later, the server does not respond with error code ERROR\_SERVICE\_DATABASE\_LOCKED (1055) for RStartServiceA (section 3.1.4.30) and RStartServiceW (section 3.1.4.19) RPCs after the database is locked using RLockServiceDatabase (section 3.1.4.4).

[<47> Section 3.1.4.30](#Appendix_A_Target_47): Windows waits 30 seconds for the service to respond.

[<48> Section 3.1.4.30](#Appendix_A_Target_48): In Windows NT, Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2, after the database is locked, the server does not allow further client operations on the database until it is unlocked. In Windows Vista and later and Windows Server 2008 and later, the server ignores the database lock.

In Windows NT 3.51, Windows NT 4.0, Windows 2000, Windows 2000 Server, Windows XP, Windows Server 2003, and Windows Server 2003 R2, the server responds with the error code ERROR\_SERVICE\_DATABASE\_LOCKED (1055) for RStartServiceA (section 3.1.4.30) and RStartServiceW (section 3.1.4.19) RPCs if the database has been locked using RLockServiceDatabase (section 3.1.4.4).

In Windows Vista and later and Windows Server 2008 and later, the server does not respond with error code ERROR\_SERVICE\_DATABASE\_LOCKED (1055) for RStartServiceA (section 3.1.4.30) and RStartServiceW (section 3.1.4.19) RPCs after the database is locked using RLockServiceDatabase (section 3.1.4.4).

[<49> Section 3.1.4.31](#Appendix_A_Target_49): If the *lpDisplayName* buffer is insufficient to hold the complete display name of the service, Windows fails the call and sets double of the size in chars of the display name excluding the terminating null character in *lpcchBuffer*.

[<50> Section 3.1.4.32](#Appendix_A_Target_50): If the *lpKeyName* buffer is insufficient to hold the complete service name of the service, Windows fails the call and sets double of the size in chars of the service name excluding the terminating null character in *lpcchBuffer*.

[<51> Section 3.1.4.34](#Appendix_A_Target_51): In Windows NT, ERROR\_CALL\_NOT\_IMPLEMENTED (120) is returned.

[<52> Section 3.1.4.34](#Appendix_A_Target_52): Windows 2000, Windows XP, Windows Server 2003, Windows Server 2003 R2, and Windows Vista return ERROR\_INVALID\_LEVEL if **psti** or **pspn** (see section [2.2.21](#Section_f8c400d3328b4e6b9d858135c8f790a4)) is specified in the *Info* parameter.

[<53> Section 3.1.4.35](#Appendix_A_Target_53): Windows returns ERROR\_CALL\_NOT\_IMPLEMENTED (120) for Windows NT.

[<54> Section 3.1.4.35](#Appendix_A_Target_54): Windows 2000, Windows XP, Windows Server 2003, Windows Server 2003 R2, and Windows Vista return ERROR\_INVALID\_LEVEL if **psti** or **pspn** (section 2.2.21) is specified in the *Info* parameter.

[<55> Section 3.1.4.36](#Appendix_A_Target_55): Windows returns ERROR\_CALL\_NOT\_IMPLEMENTED (120) for Windows NT.

[<56> Section 3.1.4.36](#Appendix_A_Target_56): ERROR\_INVALID\_PARAMETER (87) is returned in Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2.

[<57> Section 3.1.4.36](#Appendix_A_Target_57): ERROR\_INVALID\_PARAMETER (87) is returned in Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2.

[<58> Section 3.1.4.36](#Appendix_A_Target_58): ERROR\_INVALID\_PARAMETER (87) is returned in Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2.

[<59> Section 3.1.4.36](#Appendix_A_Target_59): ERROR\_INVALID\_PARAMETER (87) is returned in Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2.

[<60> Section 3.1.4.36](#Appendix_A_Target_60): Windows returns ERROR\_INVALID\_PARAMETER (87) for Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2.

[<61> Section 3.1.4.36](#Appendix_A_Target_61): ERROR\_INVALID\_PARAMETER (87) is returned in Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2.

[<62> Section 3.1.4.36](#Appendix_A_Target_62): Available in Windows 7 and Windows Server 2008 R2.

[<63> Section 3.1.4.36](#Appendix_A_Target_63): **Note** When the server is passing an invalid value for these parameters, behavior can change based on the RPC runtime check. See RPC Runtime Check Notes (section [3.2](#Section_e10c1dbde4f14083905798d80a9e06f1)).

[<64> Section 3.1.4.37](#Appendix_A_Target_64): Windows returns ERROR\_CALL\_NOT\_IMPLEMENTED (120) for Windows NT.

[<65> Section 3.1.4.37](#Appendix_A_Target_65): Windows returns ERROR\_INVALID\_PARAMETER (87) for Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2.

[<66> Section 3.1.4.37](#Appendix_A_Target_66): Windows returns ERROR\_INVALID\_PARAMETER (87) for Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2.

[<67> Section 3.1.4.37](#Appendix_A_Target_67): Windows returns ERROR\_INVALID\_PARAMETER (87) for Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2.

[<68> Section 3.1.4.37](#Appendix_A_Target_68): Windows returns ERROR\_INVALID\_PARAMETER (87) for Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2.

[<69> Section 3.1.4.37](#Appendix_A_Target_69): Windows returns ERROR\_INVALID\_PARAMETER (87) for Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2.

[<70> Section 3.1.4.37](#Appendix_A_Target_70): Windows returns ERROR\_INVALID\_PARAMETER (87) for Windows 2000, Windows XP, Windows Server 2003, and Windows Server 2003 R2.

[<71> Section 3.1.4.37](#Appendix_A_Target_71): Available in Windows 7 and Windows Server 2008 R2.

[<72> Section 3.1.4.37](#Appendix_A_Target_72): **Note** When the server is passing an invalid value for these parameters, behavior can change based on the RPC runtime check. See RPC Runtime Check Notes (section 3.2).

[<73> Section 3.1.4.41](#Appendix_A_Target_73): If the *lpBinaryPathName* has the "%windir%\System32" folder specified within the path, which is the 64-bit location on 64-bit Windows, Windows automatically replaces that folder with "%windir%\SysWow64", which is the 32-bit location on 64-bit Windows.

[<74> Section 3.1.4.43](#Appendix_A_Target_74): Available in Windows Vista, Windows Server 2008 operating system, Windows 7, and Windows Server 2008 R2.

[<75> Section 3.1.4.44](#Appendix_A_Target_75): Available in Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2.

[<76> Section 3.1.4.44](#Appendix_A_Target_76): Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2 clients use multiplexed RPC connections for RGetNotifyResults on request if the server supports them, and they fall back to non-multiplexed connections if the server doesn't support multiplexed connections.

[<77> Section 3.1.4.45](#Appendix_A_Target_77): Not available in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<78> Section 3.1.4.46](#Appendix_A_Target_78): Available in Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2.

[<79> Section 3.1.4.46](#Appendix_A_Target_79): Windows waits 30 seconds for the service to respond.

[<80> Section 3.1.4.47](#Appendix_A_Target_80): Available in Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2.

[<81> Section 3.1.4.47](#Appendix_A_Target_81): Windows waits 30 seconds for the service to respond.

[<82> Section 3.1.4.48](#Appendix_A_Target_82): This method is available only in Windows 7.

[<83> Section 6](#Appendix_A_Target_83): Windows XP does not support [range] on strings.

# Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

# Index

A

[Abstract data model](#section_3b30d63a095b4bc184ec72370fc1db61) 49

[server](#section_3b30d63a095b4bc184ec72370fc1db61) 49

[ANSI and Unicode string formats - conversion](#section_f9ddeebeb05e423c944084fd3d14bee9) 140

[Applicability](#section_8667f29db04641e7908c7c5f6d96b439) 11

C

[Capability negotiation](#section_ecdcdc9cc71346ee94d9624f3dc55b4a) 11

[Change tracking](#section_3d35566fda8c4aa997b2601e647e2e5b) 165

[Client - transport](#section_1901c4c33d6e426989cc42e21f0e9892) 13

[Common data types](#section_b22c221bc5544c72964cb88d9a780ea3) 13

[Common error codes](#section_c6fcc92f1fec485285f6feaab47b7810) 48

[Conversion between ANSI and Unicode string formats](#section_f9ddeebeb05e423c944084fd3d14bee9) 140

D

[Data model - abstract](#section_3b30d63a095b4bc184ec72370fc1db61) 49

[server](#section_3b30d63a095b4bc184ec72370fc1db61) 49

[Data types](#section_b22c221bc5544c72964cb88d9a780ea3) 13

[common - overview](#section_b22c221bc5544c72964cb88d9a780ea3) 13

E

[ENUM\_SERVICE\_STATUS\_PROCESSA structure](#section_45beeb69d09d427c945b9e8a774a0d09) 16

[ENUM\_SERVICE\_STATUS\_PROCESSW structure](#section_fe73e0f37ed148eb8265a6f8c33e3e05) 17

[ENUM\_SERVICE\_STATUSA structure](#section_867593839e734b46aeb973d21c2d1f6c) 16

[ENUM\_SERVICE\_STATUSW structure](#section_79f17326c1384fbc81705c11c4b3ca79) 16

[Error codes](#section_c6fcc92f1fec485285f6feaab47b7810) 48

Events

[local - server](#section_d5f3e3a750454ca7836f4a8c105153a1) 140

[timer - server](#section_e7fbccb8a82b42018e44d2bedce859c5) 140

[Examples](#section_b902a454b5fa403b9325029fae04cbe0) 142

[overview](#section_b902a454b5fa403b9325029fae04cbe0) 142

F

[Fields - vendor-extensible](#section_f1c23167df614ebea304f9e58a723cfb) 12

[Full IDL](#section_1916853740b54d7a99e0d77f0f5e0241) 144

G

[Glossary](#section_4f66d32e2f4f4482998ee5ccbb445223) 7

I

[IDL](#section_1916853740b54d7a99e0d77f0f5e0241) 144

[Implementer - security considerations](#section_69df313cdd564d9aba9b2641bf2965ba) 143

[Index of security parameters](#section_1a50ee6f25094a4abc4f2ffe21faabaf) 143

[Informative references](#section_6f945c52307f4fd694c012d254f427b0) 10

[Initialization](#section_cc9d037b8e40413eb35781602ecde1e3) 58

[server](#section_cc9d037b8e40413eb35781602ecde1e3) 58

[Introduction](#section_8c9ac3a251014e45b8a380282a8c3336) 7

L

[Local events](#section_d5f3e3a750454ca7836f4a8c105153a1) 140

[server](#section_d5f3e3a750454ca7836f4a8c105153a1) 140

[LPENUM\_SERVICE\_STATUS\_PROCESSA](#section_45beeb69d09d427c945b9e8a774a0d09) 16

[LPENUM\_SERVICE\_STATUS\_PROCESSW](#section_fe73e0f37ed148eb8265a6f8c33e3e05) 17

[LPENUM\_SERVICE\_STATUSA](#section_867593839e734b46aeb973d21c2d1f6c) 16

[LPENUM\_SERVICE\_STATUSW](#section_79f17326c1384fbc81705c11c4b3ca79) 16

[LPQUERY\_SERVICE\_CONFIGA](#section_daee07f590754534a6747db7fc689b36) 17

[LPQUERY\_SERVICE\_CONFIGW](#section_97200665563142ea99176f9b41f02391) 19

[LPQUERY\_SERVICE\_LOCK\_STATUSA](#section_b61893cbaef2427eabf66f3b072f629d) 21

[LPQUERY\_SERVICE\_LOCK\_STATUSW](#section_32df480ef28d4a68a18239692b7c9a78) 21

[LPSC\_ACTION](#section_5e3e9ca16c334c94bf36c5b75262d7d6) 22

[LPSERVICE\_DELAYED\_AUTO\_START\_INFO](#section_805b8296863d4d1e8ae8f639adf8c6cb) 31

[LPSERVICE\_DESCRIPTIONA](#section_91a48e76b6b843eb9ea10757c4aa9260) 32

[LPSERVICE\_DESCRIPTIONW](#section_3de0acf0bd9b4b0ba37602036b16a83f) 32

[LPSERVICE\_FAILURE\_ACTIONS\_FLAG](#section_9b244e2e82fc4c548f4fb19034faa2c4) 34

[LPSERVICE\_FAILURE\_ACTIONSA](#section_a5b207c7114e48aea1c6ef79c1fe1854) 33

[LPSERVICE\_FAILURE\_ACTIONSW](#section_58032b711e5c4f2e854534b0f2e8c6ad) 34

[LPSERVICE\_PREFERRED\_NODE\_INFO](#section_3767259ff0544076acd39f7ecb79cab3) 47

[LPSERVICE\_PRESHUTDOWN\_INFO](#section_6b55ad95df3e498fb107e77f831ba467) 37

[LPSERVICE\_RPC\_REQUIRED\_PRIVILEGES\_INFO](#section_2917121dbbe147d2ad557bce44a127e6) 40

[LPSERVICE\_SID\_INFO](#section_ea1a9acd4bb2473fae5b55969c5960fc) 38

[LPSERVICE\_STATUS](#section_4e91ff36ab5f49eda43da308e72b0b3c) 38

[LPSERVICE\_STATUS\_PROCESS](#section_c2f0ab87eb1643718380ddf9cd29931e) 41

[LPSTRING\_PTRSA](#section_2498d96ab2304023b94dc219888fd623) 43

[LPSTRING\_PTRSW](#section_8e40b857e9df4600aec4c20ca2b13dd8) 43

M

[MAX\_SERVICE\_NAME\_LENGTH](#section_15fcdce1424a4c999965629f2cd53126) 47

[Message processing](#section_0d7a70119f41470dad528535b47ac282) 58

[server](#section_0d7a70119f41470dad528535b47ac282) 58

Messages

[common data types](#section_b22c221bc5544c72964cb88d9a780ea3) 13

[data types](#section_b22c221bc5544c72964cb88d9a780ea3) 13

[overview](#section_19171539e692461597b9fd7bbffe5b3c) 13

[transport](#section_69bbbbfa6f2e449ea5bc47db341fe81b) 13

[client](#section_1901c4c33d6e426989cc42e21f0e9892) 13

[overview](#section_69bbbbfa6f2e449ea5bc47db341fe81b) 13

[server](#section_4c8b7701b043400c9350dc29cfaa5e7a) 13

Methods

[RChangeServiceConfig2A (Opnum 36)](#section_7c3a257bbbc640bcbe10c8b013876b2a) 112

[RChangeServiceConfig2W (Opnum 37)](#section_f655d914b6244ed8b55b463f17253707) 113

[RChangeServiceConfigA (Opnum 23)](#section_232575f34bfc410f8fb3ca093005434e) 92

[RChangeServiceConfigW (Opnum 11)](#section_61ea7ed0c49d4152a164b4830f16c8a4) 73

[RCloseNotifyHandle (Opnum 49)](#section_65ce8393c4a14c3791da2f65e9f0df5d) 133

[RCloseServiceHandle (Opnum 0)](#section_a2a4e17409fb4e55bad3f77c4b13245c) 63

[RControlService (Opnum 1)](#section_e1c478be117f45129b6717c20a48af97) 63

[RControlServiceExA (Opnum 50)](#section_58f8ac09776e4853988976cc459d8ea5) 134

[RControlServiceExW (Opnum 51)](#section_1c07ddac6ad54a6695c36271cd303fd3) 137

[RCreateServiceA (Opnum 24)](#section_3ab258d687b0459e8d83a2cdd8038b78) 95

[RCreateServiceW (Opnum 12)](#section_6a8ca92694774dd4b766692fab07227e) 77

[RCreateServiceWOW64A (Opnum 44)](#section_eab9a1a3181b432087cab655c6701e85) 124

[RCreateServiceWOW64W (Opnum 45)](#section_b32e905dce864ce8a66ab2f85efaca12) 128

[RDeleteService (Opnum 2)](#section_6744cdb8f1624be0bb3198996b6495be) 66

[REnumDependentServicesA (Opnum 25)](#section_e150f53356cf4d2a9e2d5dcccf3dbf61) 99

[REnumDependentServicesW (Opnum 13)](#section_6269bea89dd34092bd3367cec685d38e) 81

[REnumServiceGroupW (Opnum 35)](#section_1781d75a0232453c9c41f05c96ce51da) 110

[REnumServicesStatusA (Opnum 26)](#section_7f4be088639945c89b3a6f54df618a94) 101

[REnumServicesStatusExA (Opnum 41)](#section_f5512859cae94a109636eefeb0abd9a4) 119

[REnumServicesStatusExW (Opnum 42)](#section_fdb7df2e341e4dccad5fd16c2ac51466) 121

[REnumServicesStatusW (Opnum 14)](#section_22b4ff3d29c6481fb5988ce66a46944a) 82

[RGetNotifyResults (Opnum 48)](#section_7020f3e117b9495680dc583ea1509fe6) 133

[RGetServiceDisplayNameA (Opnum 32)](#section_298c7656529a4b91a9c36d10a891015d) 108

[RGetServiceDisplayNameW (Opnum 20)](#section_48b42c4a26b74d599b7a98533ebeb730) 90

[RGetServiceKeyNameA (Opnum 33)](#section_0d04a3c9db344248824dec8ce05655ad) 109

[RGetServiceKeyNameW (Opnum 21)](#section_ecd403dbcc754402a5fcfcda46106205) 91

[RLockServiceDatabase (Opnum 3)](#section_ff71f732e91d41898fb9a410674c63ad) 66

[RNotifyBootConfigStatus (Opnum 9)](#section_624e57ef772d45d2ab9903455879a424) 72

[RNotifyServiceStatusChange (Opnum 47)](#section_b4cb24471f8c4deea78e209bdacadea6) 131

[ROpenSCManagerA (Opnum 27)](#section_440941b1c11f444882d4b2b71cc55f5a) 103

[ROpenSCManagerW (Opnum 15)](#section_dc84adb3d51d48eb820dba1c6ca5faf2) 84

[ROpenServiceA (Opnum 28)](#section_ddd7c91fe4fb446e90530b6308c640eb) 104

[ROpenServiceW (Opnum 16)](#section_6d0a4225451b4132894d7cef7aecfd2d) 85

[RQueryServiceConfig2A (Opnum 38)](#section_89892356ac9049d7ad99fc2ffa2a2494) 114

[RQueryServiceConfig2W (Opnum 39)](#section_a11c38b617244e62aef459b78c8bae4e) 116

[RQueryServiceConfigA (Opnum 29)](#section_2119f180734740a1afaedaae8b76dd15) 105

[RQueryServiceConfigEx (Opnum 56)](#section_cbbf4eefd3e5487a919b5172d2394d98) 139

[RQueryServiceConfigW (Opnum 17)](#section_89e2d5b119cf44ca969f38eea9fe7f3c) 86

[RQueryServiceLockStatusA (Opnum 30)](#section_05ac6988644549ca84b0a5f88925076d) 106

[RQueryServiceLockStatusW (Opnum 18)](#section_edb6dd274f894099aeeea84be7957ac0) 87

[RQueryServiceObjectSecurity (Opnum 4)](#section_7f339950ce7347829e104e1c5924594e) 67

[RQueryServiceStatus (Opnum 6)](#section_cf94d915b4e140e5872ba9cb3ad09b46) 69

[RQueryServiceStatusEx (Opnum 40)](#section_e27fce9dfd4547b1bdebc05a2fd53669) 118

[RSetServiceObjectSecurity (Opnum 5)](#section_ea93548f39174626bef72f3f8fa39299) 68

[RSetServiceStatus (Opnum 7)](#section_df67cf3b1bae4359b6841b481d27a30c) 70

[RStartServiceA (Opnum 31)](#section_f15fc39185764e30b15875c306e1cba2) 106

[RStartServiceW (Opnum 19)](#section_d9be95a2cf014bdcb30f6fe4b37ada16) 88

[RUnlockServiceDatabase (Opnum 8)](#section_3456de79525049828a30debd2ea0df92) 72

N

[Normative references](#section_ba93718b35844b53a857a4912751a6bd) 10

O

[Overview (synopsis)](#section_d5bd5712fa6444bf94333651f6a5ce97) 11

P

[Parameters - security index](#section_1a50ee6f25094a4abc4f2ffe21faabaf) 143

[Preconditions](#section_72c5c14b75dd43d3aec47419a293fcf1) 11

[Prerequisites](#section_72c5c14b75dd43d3aec47419a293fcf1) 11

[Product behavior](#section_041d2a899d7d4f7991a1c336d0a668f6) 158

Protocol Details

[overview](#section_b3568539b0894e0ab339096a08f3e371) 49

[PSC\_RPC\_NOTIFY\_PARAMS\_LIST](#section_46063c6407cb4055985c75fffb259f41) 24

[PSERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSA](#section_003621fc656f41be9368324587f3904a) 26

[PSERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSW](#section_1ac2684aea98420cac337441af91282e) 29

[PSERVICE\_CONTROL\_STATUS\_REASON\_OUT\_PARAMS](#section_2679fdcc5e6a40928958a625f2a4ace7) 31

[PSERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_1](#section_e10de8b2975e4659896bf9c5bdcc42d5) 35

[PSERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_2](#section_fb661d973f844e45abbc759f19438610) 36

[PSERVICE\_TRIGGER](#section_ea3a4b5eeb284c02a5ba740cd4a6751c) 44

[PSERVICE\_TRIGGER\_INFO](#section_a56f5538d13b49bfbd7c2c5b213fc0f1) 46

[PSERVICE\_TRIGGER\_SPECIFIC\_DATA\_ITEM](#section_8c33ca40ac7d413da76871fc2c36d6bd) 43

[PSTRING\_PTRSA](#section_2498d96ab2304023b94dc219888fd623) 43

[PSTRING\_PTRSW](#section_8e40b857e9df4600aec4c20ca2b13dd8) 43

Q

[QUERY\_SERVICE\_CONFIGA structure](#section_daee07f590754534a6747db7fc689b36) 17

[QUERY\_SERVICE\_CONFIGW structure](#section_97200665563142ea99176f9b41f02391) 19

[QUERY\_SERVICE\_LOCK\_STATUSA structure](#section_b61893cbaef2427eabf66f3b072f629d) 21

[QUERY\_SERVICE\_LOCK\_STATUSW structure](#section_32df480ef28d4a68a18239692b7c9a78) 21

R

[RChangeServiceConfig2A (Opnum 36) method](#section_7c3a257bbbc640bcbe10c8b013876b2a) 112

[RChangeServiceConfig2A method](#section_7c3a257bbbc640bcbe10c8b013876b2a) 112

[RChangeServiceConfig2W (Opnum 37) method](#section_f655d914b6244ed8b55b463f17253707) 113

[RChangeServiceConfig2W method](#section_f655d914b6244ed8b55b463f17253707) 113

[RChangeServiceConfigA (Opnum 23) method](#section_232575f34bfc410f8fb3ca093005434e) 92

[RChangeServiceConfigA method](#section_232575f34bfc410f8fb3ca093005434e) 92

[RChangeServiceConfigW (Opnum 11) method](#section_61ea7ed0c49d4152a164b4830f16c8a4) 73

[RChangeServiceConfigW method](#section_61ea7ed0c49d4152a164b4830f16c8a4) 73

[RCloseNotifyHandle (Opnum 49) method](#section_65ce8393c4a14c3791da2f65e9f0df5d) 133

[RCloseNotifyHandle method](#section_65ce8393c4a14c3791da2f65e9f0df5d) 133

[RCloseServiceHandle (Opnum 0) method](#section_a2a4e17409fb4e55bad3f77c4b13245c) 63

[RCloseServiceHandle method](#section_a2a4e17409fb4e55bad3f77c4b13245c) 63

[RControlService (Opnum 1) method](#section_e1c478be117f45129b6717c20a48af97) 63

[RControlService method](#section_e1c478be117f45129b6717c20a48af97) 63

[RControlServiceExA (Opnum 50) method](#section_58f8ac09776e4853988976cc459d8ea5) 134

[RControlServiceExA method](#section_58f8ac09776e4853988976cc459d8ea5) 134

[RControlServiceExW (Opnum 51) method](#section_1c07ddac6ad54a6695c36271cd303fd3) 137

[RControlServiceExW method](#section_1c07ddac6ad54a6695c36271cd303fd3) 137

[RCreateServiceA (Opnum 24) method](#section_3ab258d687b0459e8d83a2cdd8038b78) 95

[RCreateServiceA method](#section_3ab258d687b0459e8d83a2cdd8038b78) 95

[RCreateServiceW (Opnum 12) method](#section_6a8ca92694774dd4b766692fab07227e) 77

[RCreateServiceW method](#section_6a8ca92694774dd4b766692fab07227e) 77

[RCreateServiceWOW64A (Opnum 44) method](#section_eab9a1a3181b432087cab655c6701e85) 124

[RCreateServiceWOW64A method](#section_eab9a1a3181b432087cab655c6701e85) 124

[RCreateServiceWOW64W (Opnum 45) method](#section_b32e905dce864ce8a66ab2f85efaca12) 128

[RCreateServiceWOW64W method](#section_b32e905dce864ce8a66ab2f85efaca12) 128

[RDeleteService (Opnum 2) method](#section_6744cdb8f1624be0bb3198996b6495be) 66

[RDeleteService method](#section_6744cdb8f1624be0bb3198996b6495be) 66

[References](#section_bf22836068e746dcbc1d0cdd9f7967e1) 10

[informative](#section_6f945c52307f4fd694c012d254f427b0) 10

[normative](#section_ba93718b35844b53a857a4912751a6bd) 10

[Relationship to other protocols](#section_c369346db31546788efc75ed069243e5) 11

[REnumDependentServicesA (Opnum 25) method](#section_e150f53356cf4d2a9e2d5dcccf3dbf61) 99

[REnumDependentServicesA method](#section_e150f53356cf4d2a9e2d5dcccf3dbf61) 99

[REnumDependentServicesW (Opnum 13) method](#section_6269bea89dd34092bd3367cec685d38e) 81

[REnumDependentServicesW method](#section_6269bea89dd34092bd3367cec685d38e) 81

[REnumServiceGroupW (Opnum 35) method](#section_1781d75a0232453c9c41f05c96ce51da) 110

[REnumServiceGroupW method](#section_1781d75a0232453c9c41f05c96ce51da) 110

[REnumServicesStatusA (Opnum 26) method](#section_7f4be088639945c89b3a6f54df618a94) 101

[REnumServicesStatusA method](#section_7f4be088639945c89b3a6f54df618a94) 101

[REnumServicesStatusExA (Opnum 41) method](#section_f5512859cae94a109636eefeb0abd9a4) 119

[REnumServicesStatusExA method](#section_f5512859cae94a109636eefeb0abd9a4) 119

[REnumServicesStatusExW (Opnum 42) method](#section_fdb7df2e341e4dccad5fd16c2ac51466) 121

[REnumServicesStatusExW method](#section_fdb7df2e341e4dccad5fd16c2ac51466) 121

[REnumServicesStatusW (Opnum 14) method](#section_22b4ff3d29c6481fb5988ce66a46944a) 82

[REnumServicesStatusW method](#section_22b4ff3d29c6481fb5988ce66a46944a) 82

[RGetNotifyResults (Opnum 48) method](#section_7020f3e117b9495680dc583ea1509fe6) 133

[RGetNotifyResults method](#section_7020f3e117b9495680dc583ea1509fe6) 133

[RGetServiceDisplayNameA (Opnum 32) method](#section_298c7656529a4b91a9c36d10a891015d) 108

[RGetServiceDisplayNameA method](#section_298c7656529a4b91a9c36d10a891015d) 108

[RGetServiceDisplayNameW (Opnum 20) method](#section_48b42c4a26b74d599b7a98533ebeb730) 90

[RGetServiceDisplayNameW method](#section_48b42c4a26b74d599b7a98533ebeb730) 90

[RGetServiceKeyNameA (Opnum 33) method](#section_0d04a3c9db344248824dec8ce05655ad) 109

[RGetServiceKeyNameA method](#section_0d04a3c9db344248824dec8ce05655ad) 109

[RGetServiceKeyNameW (Opnum 21) method](#section_ecd403dbcc754402a5fcfcda46106205) 91

[RGetServiceKeyNameW method](#section_ecd403dbcc754402a5fcfcda46106205) 91

[RLockServiceDatabase (Opnum 3) method](#section_ff71f732e91d41898fb9a410674c63ad) 66

[RLockServiceDatabase method](#section_ff71f732e91d41898fb9a410674c63ad) 66

[RNotifyBootConfigStatus (Opnum 9) method](#section_624e57ef772d45d2ab9903455879a424) 72

[RNotifyBootConfigStatus method](#section_624e57ef772d45d2ab9903455879a424) 72

[RNotifyServiceStatusChange (Opnum 47) method](#section_b4cb24471f8c4deea78e209bdacadea6) 131

[RNotifyServiceStatusChange method](#section_b4cb24471f8c4deea78e209bdacadea6) 131

[ROpenSCManagerA (Opnum 27) method](#section_440941b1c11f444882d4b2b71cc55f5a) 103

[ROpenSCManagerA method](#section_440941b1c11f444882d4b2b71cc55f5a) 103

[ROpenSCManagerW (Opnum 15) method](#section_dc84adb3d51d48eb820dba1c6ca5faf2) 84

[ROpenSCManagerW method](#section_dc84adb3d51d48eb820dba1c6ca5faf2) 84

[ROpenServiceA (Opnum 28) method](#section_ddd7c91fe4fb446e90530b6308c640eb) 104

[ROpenServiceA method](#section_ddd7c91fe4fb446e90530b6308c640eb) 104

[ROpenServiceW (Opnum 16) method](#section_6d0a4225451b4132894d7cef7aecfd2d) 85

[ROpenServiceW method](#section_6d0a4225451b4132894d7cef7aecfd2d) 85

[RPC runtime check notes](#section_e10c1dbde4f14083905798d80a9e06f1) 141

[RQueryServiceConfig2A (Opnum 38) method](#section_89892356ac9049d7ad99fc2ffa2a2494) 114

[RQueryServiceConfig2A method](#section_89892356ac9049d7ad99fc2ffa2a2494) 114

[RQueryServiceConfig2W (Opnum 39) method](#section_a11c38b617244e62aef459b78c8bae4e) 116

[RQueryServiceConfig2W method](#section_a11c38b617244e62aef459b78c8bae4e) 116

[RQueryServiceConfigA (Opnum 29) method](#section_2119f180734740a1afaedaae8b76dd15) 105

[RQueryServiceConfigA method](#section_2119f180734740a1afaedaae8b76dd15) 105

[RQueryServiceConfigEx (Opnum 56) method](#section_cbbf4eefd3e5487a919b5172d2394d98) 139

[RQueryServiceConfigEx method](#section_cbbf4eefd3e5487a919b5172d2394d98) 139

[RQueryServiceConfigW (Opnum 17) method](#section_89e2d5b119cf44ca969f38eea9fe7f3c) 86

[RQueryServiceConfigW method](#section_89e2d5b119cf44ca969f38eea9fe7f3c) 86

[RQueryServiceLockStatusA (Opnum 30) method](#section_05ac6988644549ca84b0a5f88925076d) 106

[RQueryServiceLockStatusA method](#section_05ac6988644549ca84b0a5f88925076d) 106

[RQueryServiceLockStatusW (Opnum 18) method](#section_edb6dd274f894099aeeea84be7957ac0) 87

[RQueryServiceLockStatusW method](#section_edb6dd274f894099aeeea84be7957ac0) 87

[RQueryServiceObjectSecurity (Opnum 4) method](#section_7f339950ce7347829e104e1c5924594e) 67

[RQueryServiceObjectSecurity method](#section_7f339950ce7347829e104e1c5924594e) 67

[RQueryServiceStatus (Opnum 6) method](#section_cf94d915b4e140e5872ba9cb3ad09b46) 69

[RQueryServiceStatus method](#section_cf94d915b4e140e5872ba9cb3ad09b46) 69

[RQueryServiceStatusEx (Opnum 40) method](#section_e27fce9dfd4547b1bdebc05a2fd53669) 118

[RQueryServiceStatusEx method](#section_e27fce9dfd4547b1bdebc05a2fd53669) 118

[RSetServiceObjectSecurity (Opnum 5) method](#section_ea93548f39174626bef72f3f8fa39299) 68

[RSetServiceObjectSecurity method](#section_ea93548f39174626bef72f3f8fa39299) 68

[RSetServiceStatus (Opnum 7) method](#section_df67cf3b1bae4359b6841b481d27a30c) 70

[RSetServiceStatus method](#section_df67cf3b1bae4359b6841b481d27a30c) 70

[RStartServiceA (Opnum 31) method](#section_f15fc39185764e30b15875c306e1cba2) 106

[RStartServiceA method](#section_f15fc39185764e30b15875c306e1cba2) 106

[RStartServiceW (Opnum 19) method](#section_d9be95a2cf014bdcb30f6fe4b37ada16) 88

[RStartServiceW method](#section_d9be95a2cf014bdcb30f6fe4b37ada16) 88

[RUnlockServiceDatabase (Opnum 8) method](#section_3456de79525049828a30debd2ea0df92) 72

[RUnlockServiceDatabase method](#section_3456de79525049828a30debd2ea0df92) 72

S

[SC\_ACTION structure](#section_5e3e9ca16c334c94bf36c5b75262d7d6) 22

[SC\_ACTION\_TYPE enumeration](#section_7e3bf33ced3d4bc68781cc9fe092de9e) 21

[SC\_ENUM\_TYPE enumeration](#section_9ee28cc619b7464b8464d645e4189e76) 22

[SC\_MAX\_ACCOUNT\_NAME\_LENGTH](#section_15fcdce1424a4c999965629f2cd53126) 47

[SC\_MAX\_ARGUMENT\_LENGTH](#section_15fcdce1424a4c999965629f2cd53126) 47

[SC\_MAX\_ARGUMENTS](#section_15fcdce1424a4c999965629f2cd53126) 47

[SC\_MAX\_COMMENT\_LENGTH](#section_15fcdce1424a4c999965629f2cd53126) 47

[SC\_MAX\_COMPUTER\_NAME\_LENGTH](#section_15fcdce1424a4c999965629f2cd53126) 47

[SC\_MAX\_DEPEND\_SIZE](#section_15fcdce1424a4c999965629f2cd53126) 47

[SC\_MAX\_NAME\_LENGTH](#section_15fcdce1424a4c999965629f2cd53126) 47

[SC\_MAX\_PATH\_LENGTH](#section_15fcdce1424a4c999965629f2cd53126) 47

[SC\_MAX\_PWD\_SIZE](#section_15fcdce1424a4c999965629f2cd53126) 47

[SC\_RPC\_CONFIG\_INFOA structure](#section_f8c400d3328b4e6b9d858135c8f790a4) 22

[SC\_RPC\_CONFIG\_INFOW [Protocol]](#section_4225730329d24ea6b4d28d5a95e4e3e0) 23

[SC\_RPC\_CONFIG\_INFOW structure](#section_4225730329d24ea6b4d28d5a95e4e3e0) 23

[SC\_RPC\_NOTIFY\_PARAMS structure](#section_0e88b912da92411d915ab9b7df6d64fc) 24

[SC\_RPC\_NOTIFY\_PARAMS\_LIST structure](#section_46063c6407cb4055985c75fffb259f41) 24

[SC\_STATUS\_TYPE enumeration](#section_a7de3a4b0b9e4b9b8863b3dbc9bbe02b) 26

Security

[implementer considerations](#section_69df313cdd564d9aba9b2641bf2965ba) 143

[overview](#section_44f685535a514bfba0160f6b25938712) 143

[parameter index](#section_1a50ee6f25094a4abc4f2ffe21faabaf) 143

[Sequencing rules](#section_0d7a70119f41470dad528535b47ac282) 58

[server](#section_0d7a70119f41470dad528535b47ac282) 58

Server

[abstract data model](#section_3b30d63a095b4bc184ec72370fc1db61) 49

[initialization](#section_cc9d037b8e40413eb35781602ecde1e3) 58

[local events](#section_d5f3e3a750454ca7836f4a8c105153a1) 140

[message processing](#section_0d7a70119f41470dad528535b47ac282) 58

[RChangeServiceConfig2A (Opnum 36) method](#section_7c3a257bbbc640bcbe10c8b013876b2a) 112

[RChangeServiceConfig2W (Opnum 37) method](#section_f655d914b6244ed8b55b463f17253707) 113

[RChangeServiceConfigA (Opnum 23) method](#section_232575f34bfc410f8fb3ca093005434e) 92

[RChangeServiceConfigW (Opnum 11) method](#section_61ea7ed0c49d4152a164b4830f16c8a4) 73

[RCloseNotifyHandle (Opnum 49) method](#section_65ce8393c4a14c3791da2f65e9f0df5d) 133

[RCloseServiceHandle (Opnum 0) method](#section_a2a4e17409fb4e55bad3f77c4b13245c) 63

[RControlService (Opnum 1) method](#section_e1c478be117f45129b6717c20a48af97) 63

[RControlServiceExA (Opnum 50) method](#section_58f8ac09776e4853988976cc459d8ea5) 134

[RControlServiceExW (Opnum 51) method](#section_1c07ddac6ad54a6695c36271cd303fd3) 137

[RCreateServiceA (Opnum 24) method](#section_3ab258d687b0459e8d83a2cdd8038b78) 95

[RCreateServiceW (Opnum 12) method](#section_6a8ca92694774dd4b766692fab07227e) 77

[RCreateServiceWOW64A (Opnum 44) method](#section_eab9a1a3181b432087cab655c6701e85) 124

[RCreateServiceWOW64W (Opnum 45) method](#section_b32e905dce864ce8a66ab2f85efaca12) 128

[RDeleteService (Opnum 2) method](#section_6744cdb8f1624be0bb3198996b6495be) 66

[REnumDependentServicesA (Opnum 25) method](#section_e150f53356cf4d2a9e2d5dcccf3dbf61) 99

[REnumDependentServicesW (Opnum 13) method](#section_6269bea89dd34092bd3367cec685d38e) 81

[REnumServiceGroupW (Opnum 35) method](#section_1781d75a0232453c9c41f05c96ce51da) 110

[REnumServicesStatusA (Opnum 26) method](#section_7f4be088639945c89b3a6f54df618a94) 101

[REnumServicesStatusExA (Opnum 41) method](#section_f5512859cae94a109636eefeb0abd9a4) 119

[REnumServicesStatusExW (Opnum 42) method](#section_fdb7df2e341e4dccad5fd16c2ac51466) 121

[REnumServicesStatusW (Opnum 14) method](#section_22b4ff3d29c6481fb5988ce66a46944a) 82

[RGetNotifyResults (Opnum 48) method](#section_7020f3e117b9495680dc583ea1509fe6) 133

[RGetServiceDisplayNameA (Opnum 32) method](#section_298c7656529a4b91a9c36d10a891015d) 108

[RGetServiceDisplayNameW (Opnum 20) method](#section_48b42c4a26b74d599b7a98533ebeb730) 90

[RGetServiceKeyNameA (Opnum 33) method](#section_0d04a3c9db344248824dec8ce05655ad) 109

[RGetServiceKeyNameW (Opnum 21) method](#section_ecd403dbcc754402a5fcfcda46106205) 91

[RLockServiceDatabase (Opnum 3) method](#section_ff71f732e91d41898fb9a410674c63ad) 66

[RNotifyBootConfigStatus (Opnum 9) method](#section_624e57ef772d45d2ab9903455879a424) 72

[RNotifyServiceStatusChange (Opnum 47) method](#section_b4cb24471f8c4deea78e209bdacadea6) 131

[ROpenSCManagerA (Opnum 27) method](#section_440941b1c11f444882d4b2b71cc55f5a) 103

[ROpenSCManagerW (Opnum 15) method](#section_dc84adb3d51d48eb820dba1c6ca5faf2) 84

[ROpenServiceA (Opnum 28) method](#section_ddd7c91fe4fb446e90530b6308c640eb) 104

[ROpenServiceW (Opnum 16) method](#section_6d0a4225451b4132894d7cef7aecfd2d) 85

[RQueryServiceConfig2A (Opnum 38) method](#section_89892356ac9049d7ad99fc2ffa2a2494) 114

[RQueryServiceConfig2W (Opnum 39) method](#section_a11c38b617244e62aef459b78c8bae4e) 116

[RQueryServiceConfigA (Opnum 29) method](#section_2119f180734740a1afaedaae8b76dd15) 105

[RQueryServiceConfigEx (Opnum 56) method](#section_cbbf4eefd3e5487a919b5172d2394d98) 139

[RQueryServiceConfigW (Opnum 17) method](#section_89e2d5b119cf44ca969f38eea9fe7f3c) 86

[RQueryServiceLockStatusA (Opnum 30) method](#section_05ac6988644549ca84b0a5f88925076d) 106

[RQueryServiceLockStatusW (Opnum 18) method](#section_edb6dd274f894099aeeea84be7957ac0) 87

[RQueryServiceObjectSecurity (Opnum 4) method](#section_7f339950ce7347829e104e1c5924594e) 67

[RQueryServiceStatus (Opnum 6) method](#section_cf94d915b4e140e5872ba9cb3ad09b46) 69

[RQueryServiceStatusEx (Opnum 40) method](#section_e27fce9dfd4547b1bdebc05a2fd53669) 118

[RSetServiceObjectSecurity (Opnum 5) method](#section_ea93548f39174626bef72f3f8fa39299) 68

[RSetServiceStatus (Opnum 7) method](#section_df67cf3b1bae4359b6841b481d27a30c) 70

[RStartServiceA (Opnum 31) method](#section_f15fc39185764e30b15875c306e1cba2) 106

[RStartServiceW (Opnum 19) method](#section_d9be95a2cf014bdcb30f6fe4b37ada16) 88

[RUnlockServiceDatabase (Opnum 8) method](#section_3456de79525049828a30debd2ea0df92) 72

[sequencing rules](#section_0d7a70119f41470dad528535b47ac282) 58

[timer events](#section_e7fbccb8a82b42018e44d2bedce859c5) 140

[timers](#section_9f5fc3c0c79e437f9f60a89726076c9a) 57

[Server - overview](#section_4c8b7701b043400c9350dc29cfaa5e7a) 13

[SERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSA structure](#section_003621fc656f41be9368324587f3904a) 26

[SERVICE\_CONTROL\_STATUS\_REASON\_IN\_PARAMSW structure](#section_1ac2684aea98420cac337441af91282e) 29

[SERVICE\_CONTROL\_STATUS\_REASON\_OUT\_PARAMS structure](#section_2679fdcc5e6a40928958a625f2a4ace7) 31

[SERVICE\_DELAYED\_AUTO\_START\_INFO structure](#section_805b8296863d4d1e8ae8f639adf8c6cb) 31

[SERVICE\_DESCRIPTION\_WOW64 structure](#section_90b40ea4c24f42a281503b5173bfc90a) 32

[SERVICE\_DESCRIPTIONA structure](#section_91a48e76b6b843eb9ea10757c4aa9260) 32

[SERVICE\_DESCRIPTIONW structure](#section_3de0acf0bd9b4b0ba37602036b16a83f) 32

[SERVICE\_FAILURE\_ACTIONS\_FLAG structure](#section_9b244e2e82fc4c548f4fb19034faa2c4) 34

[SERVICE\_FAILURE\_ACTIONS\_WOW64 structure](#section_9489030edd8249db8505b3d72c6a9271) 32

[SERVICE\_FAILURE\_ACTIONSA structure](#section_a5b207c7114e48aea1c6ef79c1fe1854) 33

[SERVICE\_FAILURE\_ACTIONSW structure](#section_58032b711e5c4f2e854534b0f2e8c6ad) 34

[SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_1 structure](#section_e10de8b2975e4659896bf9c5bdcc42d5) 35

[SERVICE\_NOTIFY\_STATUS\_CHANGE\_PARAMS\_2 structure](#section_fb661d973f844e45abbc759f19438610) 36

[SERVICE\_PREFERRED\_NODE\_INFO structure](#section_3767259ff0544076acd39f7ecb79cab3) 47

[SERVICE\_PRESHUTDOWN\_INFO structure](#section_6b55ad95df3e498fb107e77f831ba467) 37

[SERVICE\_REQUIRED\_PRIVILEGES\_INFO\_WOW64 structure](#section_99ecc0e7b2d1466c8e227f2b71786ff6) 33

[SERVICE\_RPC\_REQUIRED\_PRIVILEGES\_INFO structure](#section_2917121dbbe147d2ad557bce44a127e6) 40

[SERVICE\_SID\_INFO structure](#section_ea1a9acd4bb2473fae5b55969c5960fc) 38

[SERVICE\_STATUS structure](#section_4e91ff36ab5f49eda43da308e72b0b3c) 38

[SERVICE\_STATUS\_PROCESS structure](#section_c2f0ab87eb1643718380ddf9cd29931e) 41

[SERVICE\_TRIGGER structure](#section_ea3a4b5eeb284c02a5ba740cd4a6751c) 44

[SERVICE\_TRIGGER\_INFO structure](#section_a56f5538d13b49bfbd7c2c5b213fc0f1) 46

[SERVICE\_TRIGGER\_SPECIFIC\_DATA\_ITEM structure](#section_8c33ca40ac7d413da76871fc2c36d6bd) 43

[Standards assignments](#section_e7a38186cde240ad90c7650822bd6333) 12

[STRING\_PTRSA structure](#section_2498d96ab2304023b94dc219888fd623) 43

[STRING\_PTRSW structure](#section_8e40b857e9df4600aec4c20ca2b13dd8) 43

T

[Timer events](#section_e7fbccb8a82b42018e44d2bedce859c5) 140

[server](#section_e7fbccb8a82b42018e44d2bedce859c5) 140

[Timers](#section_9f5fc3c0c79e437f9f60a89726076c9a) 57

[server](#section_9f5fc3c0c79e437f9f60a89726076c9a) 57

[Tracking changes](#section_3d35566fda8c4aa997b2601e647e2e5b) 165

[Transport](#section_69bbbbfa6f2e449ea5bc47db341fe81b) 13

[client](#section_1901c4c33d6e426989cc42e21f0e9892) 13

[overview](#section_69bbbbfa6f2e449ea5bc47db341fe81b) 13

[server](#section_4c8b7701b043400c9350dc29cfaa5e7a) 13

U

[Unicode string formats and ANSI - conversion](#section_f9ddeebeb05e423c944084fd3d14bee9) 140

V

[Vendor-extensible fields](#section_f1c23167df614ebea304f9e58a723cfb) 12

[Versioning](#section_ecdcdc9cc71346ee94d9624f3dc55b4a) 11