**[MS-TSTS]:**

**Terminal Services Terminal Server Runtime Interface 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 |
| --- | --- | --- | --- |
| 7/20/2007 | 0.1 | Major | MCPP Milestone 5 Initial Availability |
| 9/28/2007 | 0.2 | Minor | Made changes to technical and editorial content based on feedback. |
| 10/23/2007 | 0.3 | Minor | Made technical and editorial changes based on feedback. |
| 11/30/2007 | 0.4 | Minor | Made technical and editorial changes based on feedback. |
| 1/25/2008 | 1.0 | Major | Updated and revised the technical content. |
| 3/14/2008 | 2.0 | Major | IDL files and data typing revised. |
| 5/16/2008 | 2.0.1 | Editorial | Changed language and formatting in the technical content. |
| 6/20/2008 | 3.0 | Major | Updated and revised the technical content. |
| 7/25/2008 | 4.0 | Major | Updated and revised the technical content. |
| 8/29/2008 | 5.0 | Major | Updated and revised the technical content. |
| 10/24/2008 | 6.0 | Major | Updated and revised the technical content. |
| 12/5/2008 | 7.0 | Major | Updated and revised the technical content. |
| 1/16/2009 | 7.1 | Minor | Clarified the meaning of the technical content. |
| 2/27/2009 | 8.0 | Major | Updated and revised the technical content. |
| 4/10/2009 | 8.1 | Minor | Clarified the meaning of the technical content. |
| 5/22/2009 | 8.1.1 | Editorial | Changed language and formatting in the technical content. |
| 7/2/2009 | 9.0 | Major | Updated and revised the technical content. |
| 8/14/2009 | 9.1 | Minor | Clarified the meaning of the technical content. |
| 9/25/2009 | 9.2 | Minor | Clarified the meaning of the technical content. |
| 11/6/2009 | 9.3 | Minor | Clarified the meaning of the technical content. |
| 12/18/2009 | 9.4 | Minor | Clarified the meaning of the technical content. |
| 1/29/2010 | 9.4.1 | Editorial | Changed language and formatting in the technical content. |
| 3/12/2010 | 9.5 | Minor | Clarified the meaning of the technical content. |
| 4/23/2010 | 10.0 | Major | Updated and revised the technical content. |
| 6/4/2010 | 11.0 | Major | Updated and revised the technical content. |
| 7/16/2010 | 12.0 | Major | Updated and revised the technical content. |
| 8/27/2010 | 12.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 10/8/2010 | 12.1 | Minor | Clarified the meaning of the technical content. |
| 11/19/2010 | 13.0 | Major | Updated and revised the technical content. |
| 1/7/2011 | 14.0 | Major | Updated and revised the technical content. |
| 2/11/2011 | 15.0 | Major | Updated and revised the technical content. |
| 3/25/2011 | 16.0 | Major | Updated and revised the technical content. |
| 5/6/2011 | 16.1 | Minor | Clarified the meaning of the technical content. |
| 6/17/2011 | 16.2 | Minor | Clarified the meaning of the technical content. |
| 9/23/2011 | 17.0 | Major | Updated and revised the technical content. |
| 12/16/2011 | 18.0 | Major | Updated and revised the technical content. |
| 3/30/2012 | 18.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 7/12/2012 | 18.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 10/25/2012 | 18.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 1/31/2013 | 18.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 8/8/2013 | 19.0 | Major | Updated and revised the technical content. |
| 11/14/2013 | 20.0 | Major | Updated and revised the technical content. |
| 2/13/2014 | 21.0 | Major | Updated and revised the technical content. |
| 5/15/2014 | 21.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 6/30/2015 | 22.0 | Major | Significantly changed the technical content. |
| 10/16/2015 | 23.0 | Major | Significantly changed the technical content. |
| 7/14/2016 | 24.0 | Major | Significantly changed the technical content. |
| 6/1/2017 | 24.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 9/15/2017 | 25.0 | Major | Significantly changed the technical content. |
| 12/1/2017 | 25.0 | None | No changes to the meaning, language, or formatting of the technical content. |

Table of Contents

[1 Introduction 10](#_Toc499828641)

[1.1 Glossary 10](#_Toc499828642)

[1.2 References 12](#_Toc499828643)

[1.2.1 Normative References 12](#_Toc499828644)

[1.2.2 Informative References 14](#_Toc499828645)

[1.3 Overview 14](#_Toc499828646)

[1.4 Relationship to Other Protocols 15](#_Toc499828647)

[1.5 Prerequisites/Preconditions 15](#_Toc499828648)

[1.6 Applicability Statement 15](#_Toc499828649)

[1.7 Versioning and Capability Negotiation 16](#_Toc499828650)

[1.8 Vendor-Extensible Fields 16](#_Toc499828651)

[1.9 Standards Assignments 16](#_Toc499828652)

[2 Messages 17](#_Toc499828653)

[2.1 Transport 17](#_Toc499828654)

[2.2 Common Data Types 17](#_Toc499828655)

[2.2.1 Data Types 18](#_Toc499828656)

[2.2.1.1 SESSION\_HANDLE 18](#_Toc499828657)

[2.2.1.2 ENUM\_HANDLE 18](#_Toc499828658)

[2.2.1.3 HLISTENER 18](#_Toc499828659)

[2.2.1.4 SERVER\_HANDLE 18](#_Toc499828660)

[2.2.1.5 WINSTATIONNAME 18](#_Toc499828661)

[2.2.1.6 DLLNAME 19](#_Toc499828662)

[2.2.1.7 DEVICENAME 19](#_Toc499828663)

[2.2.1.8 WINSTATIONINFOCLASS 19](#_Toc499828664)

[2.2.1.9 WINSTATIONSTATECLASS 21](#_Toc499828665)

[2.2.1.10 SDCLASS 22](#_Toc499828666)

[2.2.1.11 SHADOWCLASS 22](#_Toc499828667)

[2.2.1.12 RECONNECT\_TYPE 23](#_Toc499828668)

[2.2.1.13 CLIENTDATANAME 23](#_Toc499828669)

[2.2.1.14 TNotificationId 23](#_Toc499828670)

[2.2.1.15 NOTIFY\_HANDLE 24](#_Toc499828671)

[2.2.1.16 BOUNDED\_ULONG 25](#_Toc499828672)

[2.2.1.17 UINT\_PTR 25](#_Toc499828673)

[2.2.1.18 SESSIONTYPE 25](#_Toc499828674)

[2.2.1.19 SHADOW\_CONTROL\_REQUEST 25](#_Toc499828675)

[2.2.1.20 SHADOW\_PERMISSION\_REQUEST 26](#_Toc499828676)

[2.2.1.21 SHADOW\_REQUEST\_RESPONSE 26](#_Toc499828677)

[2.2.2 Structures 27](#_Toc499828678)

[2.2.2.1 SESSION\_FILTER 27](#_Toc499828679)

[2.2.2.2 PROTOCOLSTATUS\_INFO\_TYPE 27](#_Toc499828680)

[2.2.2.3 QUERY\_SESSION\_DATA\_TYPE 27](#_Toc499828681)

[2.2.2.4 PSESSIONENUM 28](#_Toc499828682)

[2.2.2.4.1 SessionInfo 28](#_Toc499828683)

[2.2.2.4.1.1 SESSIONENUM\_LEVEL1 29](#_Toc499828684)

[2.2.2.4.1.2 SESSIONENUM\_LEVEL2 29](#_Toc499828685)

[2.2.2.4.1.3 SESSIONENUM\_LEVEL3 29](#_Toc499828686)

[2.2.2.5 PSESSIONENUM\_EX 30](#_Toc499828687)

[2.2.2.5.1 SessionInfo\_Ex 30](#_Toc499828688)

[2.2.2.6 PEXECENVDATA 31](#_Toc499828689)

[2.2.2.6.1 ExecEnvData 31](#_Toc499828690)

[2.2.2.6.1.1 EXECENVDATA\_LEVEL1 32](#_Toc499828691)

[2.2.2.6.1.2 EXECENVDATA\_LEVEL2 32](#_Toc499828692)

[2.2.2.7 PEXECENVDATAEX 33](#_Toc499828693)

[2.2.2.7.1 ExecEnvDataEx 33](#_Toc499828694)

[2.2.2.7.1.1 EXECENVDATAEX\_LEVEL1 33](#_Toc499828695)

[2.2.2.8 PLSMSESSIONINFORMATION 34](#_Toc499828696)

[2.2.2.9 PLSMSESSIONINFORMATION\_EX 35](#_Toc499828697)

[2.2.2.10 LSM\_SESSIONINFO\_EX 35](#_Toc499828698)

[2.2.2.11 LSM\_SESSIONINFO\_EX\_LEVEL1 35](#_Toc499828699)

[2.2.2.12 PLISTENERENUM 36](#_Toc499828700)

[2.2.2.12.1 ListenerInfo 37](#_Toc499828701)

[2.2.2.12.1.1 LISTENERENUM\_LEVEL1 37](#_Toc499828702)

[2.2.2.13 LOGONID 37](#_Toc499828703)

[2.2.2.14 TS\_PROCESS\_INFORMATION\_NT4 38](#_Toc499828704)

[2.2.2.15 TS\_ALL\_PROCESSES\_INFO 38](#_Toc499828705)

[2.2.2.15.1 TS\_SYS\_PROCESS\_INFORMATION 38](#_Toc499828706)

[2.2.2.15.1.1 TS\_UNICODE\_STRING 40](#_Toc499828707)

[2.2.2.16 TS\_ALL\_PROCESSES\_INFO\_NT6 40](#_Toc499828708)

[2.2.2.16.1 TS\_SYS\_PROCESS\_INFORMATION\_NT6 41](#_Toc499828709)

[2.2.2.16.1.1 NT6\_TS\_UNICODE\_STRING 42](#_Toc499828710)

[2.2.2.16.2 SYSTEM\_THREAD\_INFORMATION 42](#_Toc499828711)

[2.2.2.16.2.1 CLIENT\_ID 43](#_Toc499828712)

[2.2.2.17 TS\_COUNTER 43](#_Toc499828713)

[2.2.2.17.1 TS\_COUNTER\_HEADER 44](#_Toc499828714)

[2.2.2.18 USERCONFIG 45](#_Toc499828715)

[2.2.2.18.1 CALLBACKCLASS 48](#_Toc499828716)

[2.2.2.18.2 APPLICATIONNAME 49](#_Toc499828717)

[2.2.2.19 WINSTATIONCLIENT 49](#_Toc499828718)

[2.2.2.19.1 TS\_TIME\_ZONE\_INFORMATION 52](#_Toc499828719)

[2.2.2.19.1.1 TS\_SYSTEMTIME 53](#_Toc499828720)

[2.2.2.20 WINSTATIONINFORMATION 54](#_Toc499828721)

[2.2.2.20.1 PROTOCOLSTATUS 55](#_Toc499828722)

[2.2.2.20.1.1 PROTOCOLSTATUSEX 56](#_Toc499828723)

[2.2.2.20.1.2 PROTOCOLCOUNTERS 56](#_Toc499828724)

[2.2.2.20.1.2.1 TSHARE\_COUNTERS 57](#_Toc499828725)

[2.2.2.20.1.3 CACHE\_STATISTICS 57](#_Toc499828726)

[2.2.2.20.1.3.1 RESERVED\_CACHE 58](#_Toc499828727)

[2.2.2.20.1.3.1.1 THINWIRECACHE 58](#_Toc499828728)

[2.2.2.20.1.3.2 TSHARE\_CACHE 58](#_Toc499828729)

[2.2.2.21 PDPARAMS 58](#_Toc499828730)

[2.2.2.22 NETWORKCONFIG 59](#_Toc499828731)

[2.2.2.23 ASYNCCONFIG 59](#_Toc499828732)

[2.2.2.23.1 MODEMNAME 60](#_Toc499828733)

[2.2.2.23.2 FLOWCONTROLCONFIG 60](#_Toc499828734)

[2.2.2.23.2.1 FLOWCONTROLCLASS 60](#_Toc499828735)

[2.2.2.23.2.2 RECEIVEFLOWCONTROLCLASS 61](#_Toc499828736)

[2.2.2.23.2.3 TRANSMITFLOWCONTROLCLASS 61](#_Toc499828737)

[2.2.2.23.3 CONNECTCONFIG 61](#_Toc499828738)

[2.2.2.23.3.1 ASYNCCONNECTCLASS 62](#_Toc499828739)

[2.2.2.24 NASICONFIG 62](#_Toc499828740)

[2.2.2.24.1 NASIUSERNAME 62](#_Toc499828741)

[2.2.2.24.2 NASIPASSWORD 62](#_Toc499828742)

[2.2.2.24.3 NASISESIONNAME 62](#_Toc499828743)

[2.2.2.24.4 NASISPECIFICNAME 63](#_Toc499828744)

[2.2.2.24.5 NASIFILESERVER 63](#_Toc499828745)

[2.2.2.25 OEMTDCONFIG 63](#_Toc499828746)

[2.2.2.26 PDCONFIG 63](#_Toc499828747)

[2.2.2.26.1 PDCONFIG2 63](#_Toc499828748)

[2.2.2.26.2 PDNAME 65](#_Toc499828749)

[2.2.2.27 WDCONFIG 65](#_Toc499828750)

[2.2.2.27.1 WDNAME 66](#_Toc499828751)

[2.2.2.27.2 WDPREFIX 66](#_Toc499828752)

[2.2.2.28 CDCONFIG 66](#_Toc499828753)

[2.2.2.28.1 CDCLASS 66](#_Toc499828754)

[2.2.2.28.2 CDNAME 67](#_Toc499828755)

[2.2.2.29 WINSTATIONCREATE 67](#_Toc499828756)

[2.2.2.30 WINSTATIONCONFIG2 67](#_Toc499828757)

[2.2.2.30.1 WINSTATIONCONFIG 67](#_Toc499828758)

[2.2.2.31 POLICY\_TS\_MACHINE 68](#_Toc499828759)

[2.2.2.32 WINSTATIONUSERTOKEN 74](#_Toc499828760)

[2.2.2.33 WINSTATIONVIDEODATA 74](#_Toc499828761)

[2.2.2.34 WINSTATIONLOADINDICATORDATA 74](#_Toc499828762)

[2.2.2.34.1 LOADFACTORTYPE 75](#_Toc499828763)

[2.2.2.35 WINSTATIONSHADOW 75](#_Toc499828764)

[2.2.2.35.1 SHADOWSTATECLASS 76](#_Toc499828765)

[2.2.2.36 WINSTATIONPRODID 76](#_Toc499828766)

[2.2.2.37 WINSTATIONREMOTEADDRESS 77](#_Toc499828767)

[2.2.2.38 ExtendedClientCredentials 77](#_Toc499828768)

[2.2.2.39 TS\_TRACE 78](#_Toc499828769)

[2.2.2.40 BEEPINPUT 80](#_Toc499828770)

[2.2.2.41 WINSTATIONCLIENTDATA 81](#_Toc499828771)

[2.2.2.42 SESSION\_CHANGE 81](#_Toc499828772)

[2.2.2.43 RCM\_REMOTEADDRESS 81](#_Toc499828773)

[2.2.2.44 CLIENT\_STACK\_ADDRESS 82](#_Toc499828774)

[2.2.2.45 VARDATA\_WIRE 82](#_Toc499828775)

[2.2.2.46 PDPARAMSWIRE 82](#_Toc499828776)

[2.2.2.47 WINSTACONFIGWIRE 83](#_Toc499828777)

[2.2.2.48 TSVIP\_SOCKADDR 83](#_Toc499828778)

[2.2.2.49 TSVIPAddress 84](#_Toc499828779)

[2.2.2.50 TSVIPSession 85](#_Toc499828780)

[2.2.2.51 WINSTATIONVALIDATIONINFORMATION 85](#_Toc499828781)

[2.2.2.52 WINSTATIONPRODUCTINFO 86](#_Toc499828782)

[2.3 Directory Service Schema Elements 86](#_Toc499828783)

[2.3.1 userParameters 87](#_Toc499828784)

[2.3.2 TSProperty 88](#_Toc499828785)

[2.3.3 Encoding PropValue Field in TSProperty Structure 92](#_Toc499828786)

[2.3.4 msTSProperty01 93](#_Toc499828787)

[3 Protocol Details 94](#_Toc499828788)

[3.1 Determining a Caller's Permissions and Access Rights 94](#_Toc499828789)

[3.1.1 Determining a Caller's Permissions 94](#_Toc499828790)

[3.1.2 Determining Whether a Caller Is SYSTEM 94](#_Toc499828791)

[3.1.3 Determining Whether a Caller Is an Administrator 94](#_Toc499828792)

[3.1.4 Determining Whether a Caller Is the Same User Who Logged onto the Session 95](#_Toc499828793)

[3.2 Local Session Manager Client Details 95](#_Toc499828794)

[3.2.1 Abstract Data Model 95](#_Toc499828795)

[3.2.2 Timers 95](#_Toc499828796)

[3.2.3 Initialization 95](#_Toc499828797)

[3.2.4 Processing Events and Sequencing Rules 95](#_Toc499828798)

[3.2.5 Timer Events 95](#_Toc499828799)

[3.2.6 Other Local Events 95](#_Toc499828800)

[3.3 Local Session Manager Server Details 95](#_Toc499828801)

[3.3.1 Abstract Data Model 95](#_Toc499828802)

[3.3.1.1 Abstract Data Types 96](#_Toc499828803)

[3.3.2 Timers 98](#_Toc499828804)

[3.3.3 Initialization 98](#_Toc499828805)

[3.3.4 Processing Events and Sequencing Rules 98](#_Toc499828806)

[3.3.4.1 TermSrvSession Methods 98](#_Toc499828807)

[3.3.4.1.1 RpcOpenSession (Opnum 0) 99](#_Toc499828808)

[3.3.4.1.2 RpcCloseSession (Opnum 1) 100](#_Toc499828809)

[3.3.4.1.3 RpcConnect (Opnum 2) 100](#_Toc499828810)

[3.3.4.1.4 RpcDisconnect (Opnum 3) 101](#_Toc499828811)

[3.3.4.1.5 RpcLogoff (Opnum 4) 102](#_Toc499828812)

[3.3.4.1.6 RpcGetUserName (Opnum 5) 102](#_Toc499828813)

[3.3.4.1.7 RpcGetTerminalName (Opnum 6) 102](#_Toc499828814)

[3.3.4.1.8 RpcGetState (Opnum 7) 103](#_Toc499828815)

[3.3.4.1.9 RpcIsSessionDesktopLocked (Opnum 8) 103](#_Toc499828816)

[3.3.4.1.10 RpcShowMessageBox (Opnum 9) 104](#_Toc499828817)

[3.3.4.1.11 RpcGetTimes (Opnum 10) 105](#_Toc499828818)

[3.3.4.1.12 RpcGetSessionCounters (Opnum 11) 106](#_Toc499828819)

[3.3.4.1.13 RpcGetSessionInformation (Opnum 12) 106](#_Toc499828820)

[3.3.4.1.14 RpcGetLoggedOnCount (Opnum 15) 107](#_Toc499828821)

[3.3.4.1.15 RpcGetSessionType (Opnum 16) 107](#_Toc499828822)

[3.3.4.1.16 RpcGetSessionInformationEx (Opnum 17) 108](#_Toc499828823)

[3.3.4.2 TermSrvNotification 108](#_Toc499828824)

[3.3.4.2.1 RpcWaitForSessionState (Opnum 0) 109](#_Toc499828825)

[3.3.4.2.2 RpcRegisterAsyncNotification (Opnum 1) 110](#_Toc499828826)

[3.3.4.2.3 RpcWaitAsyncNotification (Opnum 2) 110](#_Toc499828827)

[3.3.4.2.4 RpcUnRegisterAsyncNotification (Opnum 3) 111](#_Toc499828828)

[3.3.4.3 TermSrvEnumeration 111](#_Toc499828829)

[3.3.4.3.1 RpcOpenEnum (Opnum 0) 112](#_Toc499828830)

[3.3.4.3.2 RpcCloseEnum (Opnum 1) 113](#_Toc499828831)

[3.3.4.3.3 RpcFilterByState (Opnum 2) 113](#_Toc499828832)

[3.3.4.3.4 RpcFilterByCallersName (Opnum 3) 114](#_Toc499828833)

[3.3.4.3.5 RpcEnumAddFilter (Opnum 4) 114](#_Toc499828834)

[3.3.4.3.6 RpcGetEnumResult (Opnum 5) 114](#_Toc499828835)

[3.3.4.3.7 RpcFilterBySessionType (Opnum 6) 115](#_Toc499828836)

[3.3.4.3.8 RpcGetSessionIds (Opnum 8) 116](#_Toc499828837)

[3.3.4.3.9 RpcGetEnumResultEx (Opnum 9) 116](#_Toc499828838)

[3.3.4.3.10 RpcGetAllSessions (Opnum 10) 117](#_Toc499828839)

[3.3.4.3.11 RpcGetAllSessionsEx (Opnum 11) 118](#_Toc499828840)

[3.3.5 Timer Events 119](#_Toc499828841)

[3.3.6 Other Local Events 119](#_Toc499828842)

[3.4 TermSrv Client Details 119](#_Toc499828843)

[3.4.1 Abstract Data Model 119](#_Toc499828844)

[3.4.2 Timers 119](#_Toc499828845)

[3.4.3 Initialization 119](#_Toc499828846)

[3.4.4 Processing Events and Sequencing Rules 119](#_Toc499828847)

[3.4.5 Timer Events 119](#_Toc499828848)

[3.4.6 Other Local Events 119](#_Toc499828849)

[3.5 TermSrv Server Details 119](#_Toc499828850)

[3.5.1 Abstract Data Model 119](#_Toc499828851)

[3.5.1.1 Abstract Data Types 120](#_Toc499828852)

[3.5.2 Timers 121](#_Toc499828853)

[3.5.3 Initialization 121](#_Toc499828854)

[3.5.4 Message Processing Events and Sequencing Rules 121](#_Toc499828855)

[3.5.4.1 RCMPublic 121](#_Toc499828856)

[3.5.4.1.1 RpcGetClientData (Opnum 0) 122](#_Toc499828857)

[3.5.4.1.2 RpcGetConfigData (Opnum 1) 123](#_Toc499828858)

[3.5.4.1.3 RpcGetProtocolStatus (Opnum 2) 123](#_Toc499828859)

[3.5.4.1.4 RpcGetLastInputTime (Opnum 3) 124](#_Toc499828860)

[3.5.4.1.5 RpcGetRemoteAddress (Opnum 4) 124](#_Toc499828861)

[3.5.4.1.6 RpcGetAllListeners (Opnum 8) 125](#_Toc499828862)

[3.5.4.1.7 RpcGetSessionProtocolLastInputTime (Opnum 9) 125](#_Toc499828863)

[3.5.4.1.8 RpcGetUserCertificates (Opnum 10) 126](#_Toc499828864)

[3.5.4.1.9 RpcQuerySessionData (Opnum 11) 127](#_Toc499828865)

[3.5.4.2 RCMListener 128](#_Toc499828866)

[3.5.4.2.1 RpcOpenListener (Opnum 0) 128](#_Toc499828867)

[3.5.4.2.2 RpcCloseListener (Opnum 1) 129](#_Toc499828868)

[3.5.4.2.3 RpcStopListener (Opnum 2) 129](#_Toc499828869)

[3.5.4.2.4 RpcStartListener (Opnum 3) 130](#_Toc499828870)

[3.5.4.2.5 RpcIsListening (Opnum 4) 130](#_Toc499828871)

[3.5.5 Timer Events 130](#_Toc499828872)

[3.5.6 Other Local Events 131](#_Toc499828873)

[3.6 Legacy Client Details 131](#_Toc499828874)

[3.6.1 Abstract Data Model 131](#_Toc499828875)

[3.6.2 Timers 131](#_Toc499828876)

[3.6.3 Initialization 131](#_Toc499828877)

[3.6.4 Message Processing Events and Sequencing Rules 131](#_Toc499828878)

[3.6.5 Timer Events 131](#_Toc499828879)

[3.6.6 Other Local Events 131](#_Toc499828880)

[3.7 Legacy Server Details 131](#_Toc499828881)

[3.7.1 Abstract Data Model 131](#_Toc499828882)

[3.7.1.1 Abstract Data Types 131](#_Toc499828883)

[3.7.2 Timers 134](#_Toc499828884)

[3.7.3 Initialization 134](#_Toc499828885)

[3.7.4 Message Processing Events and Sequencing Rules 134](#_Toc499828886)

[3.7.4.1 LegacyApi 134](#_Toc499828887)

[3.7.4.1.1 RpcWinStationOpenServer (Opnum 0) 139](#_Toc499828888)

[3.7.4.1.2 RpcWinStationCloseServer (Opnum 1) 140](#_Toc499828889)

[3.7.4.1.3 RpcIcaServerPing (Opnum 2) 140](#_Toc499828890)

[3.7.4.1.4 RpcWinStationEnumerate (Opnum 3) 141](#_Toc499828891)

[3.7.4.1.5 RpcWinStationRename (Opnum 4) 142](#_Toc499828892)

[3.7.4.1.6 RpcWinStationQueryInformation (Opnum 5) 143](#_Toc499828893)

[3.7.4.1.7 RpcWinStationSetInformation (Opnum 6) 147](#_Toc499828894)

[3.7.4.1.8 RpcWinStationSendMessage (Opnum 7) 149](#_Toc499828895)

[3.7.4.1.9 RpcLogonIdFromWinStationName (Opnum 8) 151](#_Toc499828896)

[3.7.4.1.10 RpcWinStationNameFromLogonId (Opnum 9) 152](#_Toc499828897)

[3.7.4.1.11 RpcWinStationConnect (Opnum 10) 152](#_Toc499828898)

[3.7.4.1.12 RpcWinStationDisconnect (Opnum 13) 153](#_Toc499828899)

[3.7.4.1.13 RpcWinStationReset (Opnum 14) 154](#_Toc499828900)

[3.7.4.1.14 RpcWinStationShutdownSystem (Opnum 15) 155](#_Toc499828901)

[3.7.4.1.15 RpcWinStationWaitSystemEvent (Opnum 16) 156](#_Toc499828902)

[3.7.4.1.16 RpcWinStationShadow (Opnum 17) 158](#_Toc499828903)

[3.7.4.1.17 RpcWinStationBreakPoint (Opnum 29) 159](#_Toc499828904)

[3.7.4.1.18 RpcWinStationReadRegistry (Opnum 30) 160](#_Toc499828905)

[3.7.4.1.19 OldRpcWinStationEnumerateProcesses (Opnum 34) 160](#_Toc499828906)

[3.7.4.1.20 RpcWinStationEnumerateProcesses (Opnum 36) 161](#_Toc499828907)

[3.7.4.1.21 RpcWinStationTerminateProcess (Opnum 37) 162](#_Toc499828908)

[3.7.4.1.22 RpcWinStationGetAllProcesses (Opnum 43) 163](#_Toc499828909)

[3.7.4.1.23 RpcWinStationGetProcessSid (Opnum 44) 164](#_Toc499828910)

[3.7.4.1.24 RpcWinStationGetTermSrvCountersValue (Opnum 45) 165](#_Toc499828911)

[3.7.4.1.25 RpcWinStationReInitializeSecurity (Opnum 46) 166](#_Toc499828912)

[3.7.4.1.26 RpcWinStationGetLanAdapterName (Opnum 53) 166](#_Toc499828913)

[3.7.4.1.27 RpcWinStationUpdateSettings (Opnum 58) 167](#_Toc499828914)

[3.7.4.1.28 RpcWinStationShadowStop (Opnum 59) 168](#_Toc499828915)

[3.7.4.1.29 RpcWinStationCloseServerEx (Opnum 60) 169](#_Toc499828916)

[3.7.4.1.30 RpcWinStationIsHelpAssistantSession (Opnum 61) 170](#_Toc499828917)

[3.7.4.1.31 RpcWinStationGetMachinePolicy (Opnum 62) 171](#_Toc499828918)

[3.7.4.1.32 RpcWinStationCheckLoopBack (Opnum 65) 171](#_Toc499828919)

[3.7.4.1.33 RpcConnectCallback (Opnum 66) 172](#_Toc499828920)

[3.7.4.1.34 RpcWinStationGetAllProcesses\_NT6 (Opnum 70) 173](#_Toc499828921)

[3.7.4.1.35 RpcWinStationOpenSessionDirectory (Opnum 75) 174](#_Toc499828922)

[3.7.5 Timer Events 175](#_Toc499828923)

[3.7.6 Other Local Events 175](#_Toc499828924)

[3.8 Virtual IP Client Detail 175](#_Toc499828925)

[3.8.1 Abstract Data Model 175](#_Toc499828926)

[3.8.2 Timers 175](#_Toc499828927)

[3.8.3 Initialization 175](#_Toc499828928)

[3.8.4 Message Processing Events and Sequencing Rules 176](#_Toc499828929)

[3.8.5 Timer Events 176](#_Toc499828930)

[3.8.6 Other Local Events 176](#_Toc499828931)

[3.9 Virtual IP Server Detail 176](#_Toc499828932)

[3.9.1 Abstract Data Model 176](#_Toc499828933)

[3.9.1.1 Abstract Data Types 176](#_Toc499828934)

[3.9.2 Timers 176](#_Toc499828935)

[3.9.3 Initialization 176](#_Toc499828936)

[3.9.4 Message Processing Events and Sequencing Rules 176](#_Toc499828937)

[3.9.4.1 TSVIPPublic 176](#_Toc499828938)

[3.9.4.1.1 RpcGetSessionIP (Opnum 0) 177](#_Toc499828939)

[3.9.5 Timer Events 177](#_Toc499828940)

[3.9.6 Other Local Events 177](#_Toc499828941)

[3.10 SessEnv Details 177](#_Toc499828942)

[3.10.1 Abstract Data Model 177](#_Toc499828943)

[3.10.2 Timers 178](#_Toc499828944)

[3.10.3 Initialization 178](#_Toc499828945)

[3.10.4 Message Processing Events and Sequencing Rules 178](#_Toc499828946)

[3.10.4.1 SessEnvPublicRpc 178](#_Toc499828947)

[3.10.4.1.1 RpcShadow2 (Opnum 0) 178](#_Toc499828948)

[3.10.5 Timer Events 179](#_Toc499828949)

[3.10.6 Other Local Events 179](#_Toc499828950)

[4 Protocol Examples 180](#_Toc499828951)

[4.1 LSM Enumeration Example 180](#_Toc499828952)

[4.2 TermService Listener Example 182](#_Toc499828953)

[4.3 TermSrvBindSecure Example 184](#_Toc499828954)

[4.4 Legacy Example 187](#_Toc499828955)

[4.5 Encoding/Decoding Example 189](#_Toc499828956)

[5 Security 191](#_Toc499828957)

[5.1 Security Considerations for Implementers 191](#_Toc499828958)

[5.2 Index of Security Parameters 191](#_Toc499828959)

[6 Appendix A: Full IDL 192](#_Toc499828960)

[6.1 Appendix A.1: tspubrpc.idl 192](#_Toc499828961)

[6.2 Appendix A.2: rcmpublic.idl 198](#_Toc499828962)

[6.3 Appendix A.3: legacy.idl 201](#_Toc499828963)

[6.4 Appendix A.4: TSVIPRpc.idl 207](#_Toc499828964)

[6.5 Appendix A.5: winsta.h 207](#_Toc499828965)

[6.6 Appendix A.6: tsdef.h 223](#_Toc499828966)

[6.7 Appendix A.7: allproc.h 224](#_Toc499828967)

[6.8 Appendix A.8: SessEnvRpc.idl 228](#_Toc499828968)

[7 Appendix B: Product Behavior 230](#_Toc499828969)

[8 Change Tracking 243](#_Toc499828970)

[9 Index 244](#_Toc499828971)

# Introduction

This document specifies the Terminal Services Terminal Server Runtime Interface Protocol. The Terminal Services Terminal Server Runtime Interface Protocol is an RPC-based protocol used for remotely querying and configuring various aspects of a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998).

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:

**administrator**: A user who has complete and unrestricted access to the computer or [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).

**application server mode**: A mode in which [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) require a [**client access license (CAL)**](#gt_9b06b9f2-1680-41c7-b89c-e08f9fe92a25) to allow remote access to [**sessions**](#gt_0cd96b80-a737-4f06-bca4-cf9efb449d12) on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998).

**ASCII**: The American Standard Code for Information Interchange (ASCII) is an 8-bit character-encoding scheme based on the English alphabet. ASCII codes represent text in computers, communications equipment, and other devices that work with text. ASCII refers to a single 8-bit ASCII character or an array of 8-bit ASCII characters with the high bit of each character set to zero.

**client access license (CAL)**: A license required by a client user or device for accessing a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) configured in Application Server mode.

**directory service (DS)**: A service that stores and organizes information about a computer network's users and network shares, and that allows network administrators to manage users' access to the shares. See also Active Directory.

**domain**: A set of users and computers sharing a common namespace and management infrastructure. At least one computer member of the set must act as a domain controller (DC) and host a member list that identifies all members of the domain, as well as optionally hosting the Active Directory service. The domain controller provides authentication of members, creating a unit of trust for its members. Each domain has an identifier that is shared among its members. For more information, see [[MS-AUTHSOD]](%5bMS-AUTHSOD%5d.pdf#Section_953d700a57cb4cf7b0c3a64f34581cc9) section 1.1.1.5 and [[MS-ADTS]](%5bMS-ADTS%5d.pdf#Section_d243592709994c628c6d13ba31a52e1a).

**endpoint**: A network-specific address of a remote procedure call (RPC) server process for remote procedure calls. The actual name and type of the endpoint depends on the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) protocol sequence that is being used. For example, for RPC over TCP (RPC Protocol Sequence ncacn\_ip\_tcp), an endpoint might be TCP port 1025. For RPC over Server Message Block (RPC Protocol Sequence ncacn\_np), an endpoint might be the name of a [**named pipe**](#gt_34f1dfa8-b1df-4d77-aa6e-d777422f9dca). For more information, see [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824).

**GINA**: The Graphical Identification and Authentication binary. The binary loaded by logon Service, used by the Winlogon, to show the authentication user interface and to validate the user. The default [**GINA**](#gt_23aa4520-d05c-4416-9d4b-ca86905cb426) (MSGINA) can be replaced by a custom [**GINA**](#gt_23aa4520-d05c-4416-9d4b-ca86905cb426) if an [**administrator**](#gt_eb335f5b-619b-46ed-9138-06e9910108c3) wants to use its own authentication UI/methods such as fingerprint, voice recognition, and so on. For more information, see [[MSDN-GINA]](https://go.microsoft.com/fwlink/?LinkId=100300).

**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).

**handle**: Any token that can be used to identify and access an object such as a device, file, or a window.

**input method editor (IME)**: A process that maps keyboard input to phonetic components (or other language elements) that are specific to a selected language. IMEs are typically used with languages for which conventional keyboard representation is difficult or impossible. For example, East Asian languages are made up of thousands of distinct characters, which makes it impossible to show all of the characters on a single keyboard. To facilitate composition, the IME converts keystrokes into the characters of the target language (such as Japanese Katakana or Simplified Chinese).

**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.

**listener**: A session running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) that listens for incoming connection requests.

**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]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15).

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

**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 Desktop Protocol (RDP)**: A multi-channel protocol that allows a user to connect to a computer running Microsoft Terminal Services (TS). RDP enables the exchange of client and server settings and also enables negotiation of common settings to use for the duration of the connection, so that input, graphics, and other data can be exchanged and processed between client and server.

**remote procedure call (RPC)**: A communication protocol used primarily between client and server. The term has three definitions that are often used interchangeably: a runtime environment providing for communication facilities between computers (the RPC runtime); a set of request-and-response message exchanges between computers (the RPC exchange); and the single message from an RPC exchange (the RPC message). 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 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 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**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca)) 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]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) 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.

**session**: A collection of applications simultaneously running under the same Win32 subsystem.

**shell**: Part of the Windows user interface (UI) that organizes and controls user access to a wide variety of objects necessary for running applications and managing the operating system. The most numerous are the folders and files that reside on computer storage media. There are also a number of virtual objects such as network printers and other computers. The [**shell**](#gt_cf4261b3-230d-4aae-9e1e-ce90b10c09e6) organizes these objects into a hierarchical namespace and provides an API to access them.

**SYSTEM**: An account that is used by the operating system. For more information about system account access rights, see [[MSDN-LocSysAcct]](https://go.microsoft.com/fwlink/?LinkId=150267).

**terminal server**: A computer on which [**terminal services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) is running.

**terminal services (TS)**: A service on a server computer that allows delivery of applications, or the desktop itself, to various computing devices. When a user runs an application on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998), the application execution takes place on the server computer and only keyboard, mouse, and display information is transmitted over the network. Each user sees only his or her individual session, which is managed transparently by the server operating system and is independent of any other client session.

**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].

**Windows Station (WinStation)**: [**Sessions**](#gt_0cd96b80-a737-4f06-bca4-cf9efb449d12) running on the computer.

**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](https://go.microsoft.com/fwlink/?linkid=850906).

### 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-ADA1] Microsoft Corporation, "[Active Directory Schema Attributes A-L](%5bMS-ADA1%5d.pdf#Section_19528560f41e4623a406dabcfff0660f)".

[MS-ADA2] Microsoft Corporation, "[Active Directory Schema Attributes M](%5bMS-ADA2%5d.pdf#Section_e20ebc4e528540bab3bdffcb81c2783e)".

[MS-ADA3] Microsoft Corporation, "[Active Directory Schema Attributes N-Z](%5bMS-ADA3%5d.pdf#Section_4517e8353ee644d4bb95a94b6966bfb0)".

[MS-ADSC] Microsoft Corporation, "[Active Directory Schema Classes](%5bMS-ADSC%5d.pdf#Section_9abb5e97123d4da99557b353ab79b830)".

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

[MS-ERREF] Microsoft Corporation, "[Windows Error Codes](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90)".

[MS-RDPBCGR] Microsoft Corporation, "[Remote Desktop Protocol: Basic Connectivity and Graphics Remoting](%5bMS-RDPBCGR%5d.pdf#Section_5073f4ed1e9345e1b0396e30c385867c)".

[MS-RDPERP] Microsoft Corporation, "[Remote Desktop Protocol: Remote Programs Virtual Channel Extension](%5bMS-RDPERP%5d.pdf#Section_832759572d0e4c5288d11b4c998c6bec)".

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

[MS-RPCH] Microsoft Corporation, "[Remote Procedure Call over HTTP Protocol](%5bMS-RPCH%5d.pdf#Section_c0f4c9c51a614d10b8e2005378d1d212)".

[MSDN-ExitWindowsEx] Microsoft Corporation, "ExitWindowsEx function", [http://msdn.microsoft.com/en-us/library/aa376868(VS.85).aspx](https://go.microsoft.com/fwlink/?LinkId=212628)

[MSDN-MSGBOX] Microsoft Corporation, "Message Box", [http://msdn.microsoft.com/en-us/library/ms645505.aspx](https://go.microsoft.com/fwlink/?LinkId=93359)

[MSDN-PROCRIGHTS] Microsoft Corporation, "Process Security and Access Rights", [http://msdn.microsoft.com/en-us/library/ms684880(VS.85).aspx](https://go.microsoft.com/fwlink/?LinkId=208137)

[MSDN-PRVLGECNSTS] Microsoft Corporation, "Privilege Constants", [http://msdn.microsoft.com/en-us/library/bb530716(v=VS.85).aspx](https://go.microsoft.com/fwlink/?LinkId=212627)

[MSDN-RCMWin32\_TSRCS] Microsoft Corporation, "RemoteControl Method of the Win32\_TSRemoteControlSetting Class", [http://msdn.microsoft.com/en-us/library/aa383818(v=VS.85).aspx](https://go.microsoft.com/fwlink/?LinkId=208353)

[MSDN-RPCBIND] Microsoft Corporation, "Creating a Binding Handle", [http://msdn.microsoft.com/en-us/library/aa373609.aspx](https://go.microsoft.com/fwlink/?LinkId=93771)

[MSDN-TDIADDRESS] Microsoft Corporation, "TDI\_ADDRESS\_IP structure", [http://msdn.microsoft.com/en-us/library/ff565072.aspx](https://go.microsoft.com/fwlink/?LinkId=93316)

[MSDN-TOKENRIGHTS] Microsoft Corporation, "Access Rights for Access-Token Objects", [http://msdn.microsoft.com/en-us/library/aa374905(VS.85).aspx](https://go.microsoft.com/fwlink/?LinkId=208138)

[MSFT-SDLBTS] Microsoft Corporation, "Session Directory and Load Balancing Using Terminal Server", September 2002, [http://download.microsoft.com/download/8/6/2/8624174c-8587-4a37-8722-00139613a5bc/TS\_Session\_Directory.doc](https://go.microsoft.com/fwlink/?LinkId=90204)

[MSFT-WINMCE] Microsoft Corporation, "Windows Media Center", [http://www.microsoft.com/windows/windows-media-center/get-started/default.aspx](https://go.microsoft.com/fwlink/?LinkId=208408)

[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)

[WININTERNALS] Russinovich, M., and Solomon, D., "Microsoft Windows Internals, Fourth Edition", Microsoft Press, 2005, ISBN: 0735619174.

[X509] ITU-T, "Information Technology - Open Systems Interconnection - The Directory: Public-Key and Attribute Certificate Frameworks", Recommendation X.509, August 2005, [http://www.itu.int/rec/T-REC-X.509/en](https://go.microsoft.com/fwlink/?LinkId=90590)

### Informative References

[MSDN-DWM] Microsoft Corporation, "Desktop Window Manager", [http://msdn.microsoft.com/en-us/library/aa969540.aspx](https://go.microsoft.com/fwlink/?LinkId=122186)

[MSDN-MSGBeep] Microsoft Corporation, "MessageBeep function", [http://msdn.microsoft.com/en-us/library/ms680356(VS.85).aspx](https://go.microsoft.com/fwlink/?LinkId=128440)

[MSDN-ProductID] Microsoft Corporation, "ProductID property", [http://msdn.microsoft.com/en-us/library/aa370855(VS.85).aspx](https://go.microsoft.com/fwlink/?LinkId=207862)

[MSDN-SERIAL] Microsoft Corporation, "Serial Communications in Win32", [http://msdn.microsoft.com/en-us/library/ms810467.aspx](https://go.microsoft.com/fwlink/?LinkId=100301)

[MSDN-SOCKADDR\_IN6] Microsoft Corporation, "SOCKADDR\_IN6 structure", [http://msdn.microsoft.com/en-us/library/ff570824(VS.85).aspx](https://go.microsoft.com/fwlink/?LinkId=206745)

[MSDN-SOCKET] Microsoft Corporation, "socket function", [http://msdn.microsoft.com/en-us/library/ms740506.aspx](https://go.microsoft.com/fwlink/?LinkId=93442)

[MSDN-SYSTIME] Microsoft Corporation, "SYSTEMTIME structure", [http://msdn.microsoft.com/en-us/library/ms189104.aspx](https://go.microsoft.com/fwlink/?LinkId=93364)

[MSDN-Win32\_TSAcct] Microsoft Corporation, "Win32\_TSAccount class", [http://msdn.microsoft.com/en-us/library/aa383773(VS.85).aspx](https://go.microsoft.com/fwlink/?LinkId=119682)

[MSFT-IME] Microsoft Corporation, "Input method editors (IMEs)", [http://windows.microsoft.com/en-us/windows-8/input-method-editors](https://go.microsoft.com/fwlink/?LinkId=100302)

[MSFT-VS] Microsoft Corporation, "Virtual Server", [http://www.microsoft.com/windowsserversystem/virtualserver/default.aspx](https://go.microsoft.com/fwlink/?LinkId=123460)

[MSFT-W2KDDK] Microsoft Press, "Microsoft Windows 2000 Driver Development Reference Kit, volumes 1-3", March 2000, ISBN: 0735609292.

**Note** The Windows 2000 DDK publication is cited as it was the last DDK (Driver Development Kit) that was physically in print. All driver development documentation since then has been delivered in soft format, and is available for download here: http://www.m...

[MSFT-WINSYSINTERNALS] Microsoft Corporation, "Windows Sysinternals", [http://technet.microsoft.com/en-us/sysinternals](https://go.microsoft.com/fwlink/?LinkId=208768)

[MSFT-WSTSL] Microsoft Corporation, "Overview of Remote Desktop Licensing", [http://technet.microsoft.com/en-us/library/cc725933.aspx](https://go.microsoft.com/fwlink/?LinkId=93371)

## Overview

The Terminal Services Terminal Server Runtime Interface Protocol is a simple request-response RPC-based protocol used for remotely querying and configuring various aspects of a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). For example, this protocol can be used to query the number of active [**sessions**](#gt_0cd96b80-a737-4f06-bca4-cf9efb449d12) running on a terminal server. For every method that the server receives, it executes the method and returns a completion. The client simply returns the completion status to the caller.

The protocol consists of four major subcomponents:

* **Local Session Manager (LSM):** A system component that creates, destroys, and manages sessions.
* **TermService:** A service running on the system that manages remote connection requests.
* **VM Host Agent:** A service running on the system that monitors the states of the sessions within virtual machines hosted on the server [[MSFT-VS]](https://go.microsoft.com/fwlink/?LinkId=123460).[<1>](#Appendix_A_1" \o "Product behavior note 1)
* **Virtual IP:** A service running on the system that assigns IP addresses to sessions that are created by using remote connection requests.

The protocol can be further divided into the following functional categories:[<2>](#Appendix_A_2" \o "Product behavior note 2)

Functional categories associated with the **Local Session Manager (LSM)** subcomponent:

* **Local Session Manager (LSM) Session:** These calls collect information, and control and configure sessions running on the terminal server.
* **Local Session Manager (LSM) Notification:** These [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) calls are asynchronous and can be used to receive event notifications from the LSM.
* **Local Session Manager (LSM) Enumeration:** These calls are used to enumerate information related to sessions running on a terminal server.

Functional categories associated with the **VM Host Agent** subcomponent:

* **VM Host Agent Session:** These calls collect information as well as control and configure sessions running on the virtual machines hosted on the server.
* **VM Host Agent Notification:** These RPC calls are asynchronous and can be used to receive event notifications from **VM Host Agent**.
* **VM Host Agent Enumeration:** These calls are used to enumerate information related to sessions running on the virtual machines hosted on the server.

Functional categories associated with the **TermService** subcomponent:

* **TermService:** These calls can be used to query and configure various aspects of the TermServices running on the terminal server.
* **TermService Listener:** These calls are specific to the listener session running on the terminal server and listening for incoming connection requests.
* **Legacy:** The legacy calls used by [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) clients.

## Relationship to Other Protocols

The Terminal Services Terminal Server Runtime Interface Protocol is dependent upon [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) for its transport. This protocol uses RPC over [**named pipes**](#gt_34f1dfa8-b1df-4d77-aa6e-d777422f9dca) as specified in section [2.1](#Section_ac0535d0a8474386ae3ee93a50ff2484).

## Prerequisites/Preconditions

The Terminal Services Terminal Server Runtime Interface Protocol is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) interface and as a result has the prerequisites specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) as being common to RPC interfaces.

It is assumed that a Terminal Services Terminal Server Runtime Interface Protocol client has obtained the name of a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) that supports the Terminal Services Terminal Server Runtime Interface Protocol before this protocol is invoked. The manner in which a client obtains the terminal server name is implementation-specific.

## Applicability Statement

The Terminal Services Terminal Server Runtime Interface Protocol is appropriate only for querying and configuring a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998).

## Versioning and Capability Negotiation

None.

## Vendor-Extensible Fields

The Terminal Services Terminal Server Runtime Interface Protocol uses Win32 error codes defined in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). Vendors SHOULD reuse those values with their indicated meaning.[<3>](#Appendix_A_3" \o "Product behavior note 3) Choosing any other value might cause a collision in the future.

## Standards Assignments

In the following table, interfaces are based on binding and [**named pipes**](#gt_34f1dfa8-b1df-4d77-aa6e-d777422f9dca) are based on the RPC Programming Model Overview as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 2.

| Description | Interface UUID | Named pipe |
| --- | --- | --- |
| LSM Session (tspubrpc.idl)[<4>](#Appendix_A_4" \o "Product behavior note 4) | { 484809d6-4239-471b-b5bc-61df8c23ac48 } | \PIPE\LSM\_API\_service  \PIPE\UNIFIED\_API\_service[<5>](#Appendix_A_5" \o "Product behavior note 5) |
| LSM Notification (tspubrpc.idl)[<6>](#Appendix_A_6" \o "Product behavior note 6) | { 11899a43-2b68-4a76-92e3-a3d6ad8c26ce } | \PIPE\LSM\_API\_service  \PIPE\UNIFIED\_API\_service[<7>](#Appendix_A_7" \o "Product behavior note 7) |
| LSM Enumeration (tspubrpc.idl)[<8>](#Appendix_A_8" \o "Product behavior note 8) | { 88143fd0-c28d-4b2b-8fef-8d882f6a9390 } | \PIPE\LSM\_API\_service  \PIPE\UNIFIED\_API\_service[<9>](#Appendix_A_9" \o "Product behavior note 9) |
| TermSrv (RCMPublic.idl)[<10>](#Appendix_A_10" \o "Product behavior note 10) | { bde95fdf-eee0-45de-9e12-e5a61cd0d4fe } | \PIPE\TermSrv\_API\_service |
| TermSrv Listener (RCMPublic.idl)[<11>](#Appendix_A_11" \o "Product behavior note 11) | { 497d95a6-2d27-4bf5-9bbd-a6046957133c } | \PIPE\TermSrv\_API\_service |
| Legacy (Legacy.idl) | { 5ca4a760-ebb1-11cf-8611-00a0245420ed } | \PIPE\Ctx\_WinStation\_API\_service |
| TSVIPPublic (TSVIPRpc.idl) | {53b46b02-c73b-4a3e-8dee-b16b80672fc0} | \PIPE\TSVIP\_Service |
| SessEnvPublicRpc (SessEnvRpc.idl) | {1257B580-CE2F-4109-82D6-A9459D0BF6BC} | \PIPE\SessEnvPublicRpc |

# Messages

## Transport

The Terminal Services Terminal Server Runtime Interface Protocol uses the [**RPC protocol sequences**](#gt_0c171cc7-e9c4-41b6-95a9-536db0042c7a) as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 2.1.1 and [[MS-RPCH]](%5bMS-RPCH%5d.pdf#Section_c0f4c9c51a614d10b8e2005378d1d212) section 1.4.

The Terminal Services Terminal Server Runtime Interface Protocol uses the following static [**endpoints**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee) in addition to [**well-known endpoints**](#gt_a7498f8e-e85d-473b-9fc9-d2fffdf71c8a). These endpoints are ports for [MS-RPCH] section 1.5 and [MS-RPCE] section 2.1 on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998).

Port 3389: This endpoint is used by the terminal server to listen for incoming [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) method calls. The authenticated RPC interface allows RPC to negotiate the use of authentication and the authentication level on behalf of the [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) client and target server.

Both types of endpoints (Static endpoints and well-known endpoints) MUST be supported. The Terminal Services Terminal Server Runtime Interface Protocol MUST use the [**universally unique identifier (UUID)**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) as specified in section [1.9](#Section_4f5ff56828874c508b3638bb85170b58).

## Common Data Types

In addition to [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) base types specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) and in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15), this document uses the following definitions, as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2):

* BOOL
* BOOLEAN
* BYTE
* CHAR
* DWORD
* HANDLE
* HRESULT
* hyper
* LONG
* UCHAR
* UINT
* ULONG
* USHORT
* VOID
* WCHAR

The document also uses the following definitions:

* Some data structures described in this document, for example [USERCONFIG](#Section_dba750b8cb354e889811e2a1f8a10701) and [WINSTATIONCLIENT](#Section_673d8ac0f55748cb98a649925160d729), use ULONG members to store sets of BOOLEAN flags defined using C bit-field syntax.
* The variation in the use of data types is a result of the as-built interface.

The data types in the following sections are defined in the [**Microsoft Interface Definition Language (MIDL)**](#gt_9c5903c1-1477-4181-b451-3ba1e34a0c0c) specification for this RPC interface, as specified in section [6](#Section_c43addc7eebc491b9b012587262675e8).

### Data Types

#### SESSION\_HANDLE

This type is declared as follows:

1. typedef [context\_handle] void\* SESSION\_HANDLE;

A [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to a session on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). It is returned by [RpcOpenSession](#Section_56ede88b913b49119540389fddf879e5).

#### ENUM\_HANDLE

This type is declared as follows:

1. typedef [context\_handle] void\* ENUM\_HANDLE;

A [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) representing the session enumeration object on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). It is returned by [RpcOpenEnum](#Section_f17f4b8d8d524538acf3cdfbe581a99b).

#### HLISTENER

This type is declared as follows:

1. typedef [context\_handle] void\* HLISTENER;

A [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) representing a [**listener**](#gt_1e5aa0c2-4617-4d7a-969e-a7091c415c8d) running on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998).

#### SERVER\_HANDLE

This type is declared as follows:

1. typedef [context\_handle] void\* SERVER\_HANDLE;

A [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) returned by [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a) on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998).

#### WINSTATIONNAME

An array of WCHAR (**WINSTATIONNAME**) characters that represent the name of a session.

This type is declared as follows:

1. typedef WCHAR WINSTATIONNAME[WINSTATIONNAME\_LENGTH + 1];

#### DLLNAME

The **DLLNAME** type contains the name of a DLL.

1. typedef WCHAR DLLNAME[DLLNAME\_LENGTH + 1];
2. typedef WCHAR\* PDLLNAME;

#### DEVICENAME

The **DEVICENAME** type contains the name of a device.

1. typedef WCHAR DEVICENAME[DEVICENAME\_LENGTH + 1];
2. typedef WCHAR\* PDEVICENAME;

#### WINSTATIONINFOCLASS

The **WINSTATIONINFOCLASS** enumeration is used by [RpcWinStationQueryInformation](#Section_1bba9ff271d349a3bb262e5f6fcab3ee) and [RpcWinStationSetInformation](#Section_2a5ee131a1dd44c7988098df708061ea) to indicate the class of data for which to either query or set on the server. A brief description of each info class is appended to each enum value. See RpcWinStationQueryInformation for information about classes of data that can be queried and RpcWinStationSetInformation for classes of data that can be set.

The enum value **WinStationUnused1** MAY be used.[<12>](#Appendix_A_12" \o "Product behavior note 12)

1. typedef enum \_WINSTATIONINFOCLASS
2. {
3. WinStationCreateData,
4. WinStationConfiguration,
5. WinStationPdParams,
6. WinStationWd,
7. WinStationPd,
8. WinStationPrinter,
9. WinStationClient,
10. WinStationModules,
11. WinStationInformation,
12. WinStationTrace,
13. WinStationBeep,
14. WinStationEncryptionOff,
15. WinStationEncryptionPerm,
16. WinStationNtSecurity,
17. WinStationUserToken,
18. WinStationUnused1,
19. WinStationVideoData,
20. WinStationInitialProgram,
21. WinStationCd,
22. WinStationSystemTrace,
23. WinStationVirtualData,
24. WinStationClientData,
25. WinStationSecureDesktopEnter,
26. WinStationSecureDesktopExit,
27. WinStationLoadBalanceSessionTarget,
28. WinStationLoadIndicator,
29. WinStationShadowInfo,
30. WinStationDigProductId,
31. WinStationLockedState,
32. WinStationRemoteAddress,
33. WinStationIdleTime,
34. WinStationLastReconnectType,
35. WinStationDisallowAutoReconnect,
36. WinStationUnused2,
37. WinStationUnused3,
38. WinStationUnused4,
39. WinStationUnused5,
40. WinStationReconnectedFromId,
41. WinStationEffectsPolicy,
42. WinStationType,
43. WinStationInformationEx
44. } WINSTATIONINFOCLASS;

**WinStationCreateData:** Query the data used to create an instance of [**WinStation**](#gt_8cc89bb9-be94-4672-8a5d-9947b91bc952).[<13>](#Appendix_A_13" \o "Product behavior note 13)

**WinStationConfiguration:** Query or set the WinStation parameters.

**WinStationPdParams:** Query or set the Protocol Drivers (PD) parameters.

**WinStationWd:** Query the Window Driver (WD) configuration. (Only one WD configuration can be loaded.)

**WinStationPd:** Query the PD configuration. (Many PD configurations can be loaded).

**WinStationPrinter:** Query or set the Line Printer Terminal (LPT) mapping to printer queues.[<14>](#Appendix_A_14" \o "Product behavior note 14)

**WinStationClient:** Query information about the client.

**WinStationModules:** Query information about all client modules.

**WinStationInformation:** Query information about the WinStation.

**WinStationTrace:** Enable or disable WinStation tracing.[<15>](#Appendix_A_15" \o "Product behavior note 15)

**WinStationBeep:** Sound a beep in the WinStation.[<16>](#Appendix_A_16" \o "Product behavior note 16)

**WinStationEncryptionOff:** Turn off encryption.[<17>](#Appendix_A_17" \o "Product behavior note 17)

**WinStationEncryptionPerm:** Encryption is permanently on.[<18>](#Appendix_A_18" \o "Product behavior note 18)

**WinStationNtSecurity:** Select logon service[<19>](#Appendix_A_19" \o "Product behavior note 19) security desktop.[<20>](#Appendix_A_20" \o "Product behavior note 20)

**WinStationUserToken:** Query the primary access token of the logged-on user.

**WinStationUnused1:** Not used.

**WinStationVideoData:** Query the horizontal resolution, vertical resolution, and color depth.[<21>](#Appendix_A_21" \o "Product behavior note 21)

**WinStationInitialProgram:** Identify the initial program run by [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) when the user logs on.[<22>](#Appendix_A_22" \o "Product behavior note 22)

**WinStationCd:** Query the Client Device (CD) configuration. (Only one CD configuration can be loaded.)[<23>](#Appendix_A_23" \o "Product behavior note 23)

**WinStationSystemTrace:** Enable or disable system tracing.[<24>](#Appendix_A_24" \o "Product behavior note 24)

**WinStationVirtualData:** Query the client virtual data.

**WinStationClientData:** Send data to a client.[<25>](#Appendix_A_25" \o "Product behavior note 25)

**WinStationSecureDesktopEnter:** Turn encryption on, if enabled.[<26>](#Appendix_A_26" \o "Product behavior note 26)

**WinStationSecureDesktopExit:** Turn encryption off, if enabled.[<27>](#Appendix_A_27" \o "Product behavior note 27)

**WinStationLoadBalanceSessionTarget:** Load balance information from a redirected client.[<28>](#Appendix_A_28" \o "Product behavior note 28)

**WinStationLoadIndicator:** Query load capacity information.

**WinStationShadowInfo:** Query or set shadow state and parameters.[<29>](#Appendix_A_29" \o "Product behavior note 29)

**WinStationDigProductId:** Get the **WINSTATIONPRODID**, as specified in section [2.2.2.36](#Section_42e13db86eb349d3a66762500e568db5).[<30>](#Appendix_A_30" \o "Product behavior note 30)

**WinStationLockedState:** Used by the logon service[<31>](#Appendix_A_31" \o "Product behavior note 31) to notify applications and services.

**WinStationRemoteAddress:** Query the client IP address.

**WinStationIdleTime:** Query for the amount of time the WinStation is idle.[<32>](#Appendix_A_32" \o "Product behavior note 32)

**WinStationLastReconnectType:** Query if the last reconnect for this WinStation was manual or auto-reconnect.[<33>](#Appendix_A_33" \o "Product behavior note 33)

**WinStationDisallowAutoReconnect:** Allow or disallow auto-reconnect for this WinStation.[<34>](#Appendix_A_34" \o "Product behavior note 34)

**WinStationUnused2:** Not used.

**WinStationUnused3:** Not used.

**WinStationUnused4:** Not used.

**WinStationUnused5:** Not used.

**WinStationReconnectedFromId:** In the case of reconnected sessions, return the session ID of the temporary session from which it was reconnected, or -1 if no temporary session was created.

**WinStationEffectsPolicy:** Return policies that differentiate among implementations.

**WinStationType:** Return the type of the session associated with this WinStation.[<35>](#Appendix_A_35" \o "Product behavior note 35)

**WinStationInformationEx:** Return the extended information about the WinStation.[<36>](#Appendix_A_36" \o "Product behavior note 36)

#### WINSTATIONSTATECLASS

The WINSTATIONSTATECLASS enumeration represents the current state of a session.

1. typedef enum \_WINSTATIONSTATECLASS
2. {
3. State\_Active = 0,
4. State\_Connected = 1,
5. State\_ConnectQuery = 2,
6. State\_Shadow = 3,
7. State\_Disconnected = 4,
8. State\_Idle = 5,
9. State\_Listen = 6,
10. State\_Reset = 7,
11. State\_Down = 8,
12. State\_Init = 9
13. } WINSTATIONSTATECLASS;

**State\_Active:** A user is logged on to a session and the client is connected.

**State\_Connected:** A client is connected to a session but the user has not yet logged on.

**State\_ConnectQuery:** A session is in the process of connecting to a client.

**State\_Shadow:** A session is shadowing another session.

**State\_Disconnected:** A user is logged on to the session but the client is currently disconnected from the server.

**State\_Idle:** A session is waiting for a client to connect to the server.

**State\_Listen:** A [**listener**](#gt_1e5aa0c2-4617-4d7a-969e-a7091c415c8d) is waiting for connections from the Terminal Services client.

**State\_Reset:** A session is being reset. As a result, the user is logged off, the session is terminated, and the client is disconnected.[<37>](#Appendix_A_37" \o "Product behavior note 37)

**State\_Down:** A session is currently tearing down or is in the down state, indicating an error.

**State\_Init:** A session is in the process of being initialized.

#### SDCLASS

The **SDCLASS** (stack driver class) enumeration is used to specify a type of binary or driver in the union [PDPARAMS](#Section_ff0e2998fb8b4dffab648427fad556eb) and to indicate which structure in the union PDPARAMS applies to a given instance of the PDPARAMS structure.

1. typedef enum \_SDCLASS
2. {
3. SdNone = 0,
4. SdConsole,
5. SdNetwork,
6. SdAsync,
7. SdOemTransport
8. } SDCLASS;

**SdNone:** None.

**SdConsole:** Not used.

**SdNetwork:** Indicates the networking binaries.[<38>](#Appendix_A_38" \o "Product behavior note 38)

**SdAsync:** Indicates the async (modem) drivers.[<39>](#Appendix_A_39" \o "Product behavior note 39)

**SdOemTransport:** Indicates the user transport drivers.[<40>](#Appendix_A_40" \o "Product behavior note 40)

#### SHADOWCLASS

The **SHADOWCLASS** enumeration is used to indicate the shadow-related settings for a session running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998).

1. typedef enum \_SHADOWCLASS
2. {
3. Shadow\_Disable,
4. Shadow\_EnableInputNotify,
5. Shadow\_EnableInputNoNotify,
6. Shadow\_EnableNoInputNotify,
7. Shadow\_EnableNoInputNoNotify,
8. } SHADOWCLASS;

**Shadow\_Disable:** Shadowing is disabled.

**Shadow\_EnableInputNotify:** Permission is asked first from the session being shadowed. The shadower is also permitted keyboard and mouse input.

**Shadow\_EnableInputNoNotify:** Permission is not asked first from the session being shadowed. The shadower is also permitted keyboard and mouse input.

**Shadow\_EnableNoInputNotify:** Permission is asked first from the session being shadowed. The shadower is not permitted keyboard and mouse input and MUST observe the shadowed session.

**Shadow\_EnableNoInputNoNotify:** Permission is not asked first from the session being shadowed. The shadower is not permitted keyboard and mouse input and MUST observe the shadowed session.

#### RECONNECT\_TYPE

The RECONNECT\_TYPE enumeration specifies the reconnect type of the last session reconnect.

1. typedef enum \_RECONNECT\_TYPE
2. {
3. NeverReconnected = 0,
4. ManualReconnect = 1,
5. AutoReconnect = 2
6. } RECONNECT\_TYPE,
7. \*PRECONNECT\_TYPE;

**NeverReconnected:** Session has never been reconnected to. This is the default type until the first time the session has been reconnected to.

**ManualReconnect:** Session was disconnected from and was manually reconnected to by the user.

**AutoReconnect:** Session was disconnected from and was automatically reconnected to by the [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) client and the server negotiating the reconnect without input from the user.

#### CLIENTDATANAME

The **CLIENTDATANAME** type specifies the name of the client data being provided.

1. typedef CHAR CLIENTDATANAME[CLIENTDATANAME\_LENGTH + 1];
2. typedef CHAR \* PCLIENTDATANAME;

The name has the following form:

1. name syntax: xxxyyyy<null>

The elements are as follows:

**xxx:**  The OEM ID.

**yyyy:** Client data name.

**<null>:**  Trailing null.

#### TNotificationId

Specifies the type of notification for which to wait from the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998).

It MUST be a bitwise OR of any of the values shown in the following table.

This type is declared as follows:

1. typedef ULONG TNotificationId;

| Value | Meaning |
| --- | --- |
| WTS\_NOTIFY\_NONE  0x0 | No notification |
| WTS\_NOTIFY\_CREATE  0x1 | Session creation notification |
| WTS\_NOTIFY\_CONNECT  0x2 | Session connection notification |
| WTS\_NOTIFY\_DISCONNECT  0x4 | Session disconnection notification |
| WTS\_NOTIFY\_LOGON  0x8 | Session logon notification |
| WTS\_NOTIFY\_LOGOFF  0x10 | Session logoff notification |
| WTS\_NOTIFY\_SHADOW\_START  0x20 | Session shadow start notification |
| WTS\_NOTIFY\_SHADOW\_STOP  0x40 | Session shadow stop notification |
| WTS\_NOTIFY\_TERMINATE  0x80 | Session termination notification |
| WTS\_NOTIFY\_CONSOLE\_CONNECT  0x100 | Console session connection notification |
| WTS\_NOTIFY\_CONSOLE\_DISCONNECT  0x200 | Console session disconnect notification |
| WTS\_NOTIFY\_LOCK  0x400 | Session lock notification |
| WTS\_NOTIFY\_UNLOCK  0x800 | Session unlock notification |
| WTS\_NOTIFY\_ALL  0xffffffff | All notifications |

#### NOTIFY\_HANDLE

A [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to a notification object.

Used in asynchronous calls such as [RpcRegisterAsyncNotification](#Section_bb3d1c58f397472e82ed44c242846464) and [RpcWaitAsyncNotification](#Section_81d5ba342ae446149aca8deb1eb7a0a1).

This type is declared as follows:

1. typedef [context\_handle] void\* NOTIFY\_HANDLE;

#### BOUNDED\_ULONG

A bounded ULONG.

This type is declared as follows:

1. typedef [range(0, 0x8000)] ULONG BOUNDED\_ULONG;

#### UINT\_PTR

An unsigned integer, whose length is dependent on processor word size.

1. #if defined(\_WIN64)
2. typedef unsigned \_\_int64 UINT\_PTR;
3. #else
4. typedef unsigned int UINT\_PTR;
5. #endif

#### SESSIONTYPE

The **SESSIONTYPE** enumeration defines the type of the session.

1. typedef enum \_SessionType
2. {
3. SESSIONTYPE\_UNKNOWN = 0,
4. SESSIONTYPE\_SERVICES,
5. SESSIONTYPE\_LISTENER,
6. SESSIONTYPE\_REGULARDESKTOP,
7. SESSIONTYPE\_ALTERNATESHELL,
8. SESSIONTYPE\_REMOTEAPP,
9. SESSIONTYPE\_MEDIACENTEREXT
10. } SESSIONTYPE;

**SESSIONTYPE\_UNKNOWN:** The type of the session cannot be determined.

**SESSIONTYPE\_SERVICES:** The session is used only to run the operating system services, and that no user can be logged on to the session.

**SESSIONTYPE\_LISTENER:** The session is used only to run the [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) [**listeners**](#gt_1e5aa0c2-4617-4d7a-969e-a7091c415c8d), and that no user can be logged on to the session.

**SESSIONTYPE\_REGULARDESKTOP:** The session is connected by using Terminal Services and is running the standard [**shell**](#gt_cf4261b3-230d-4aae-9e1e-ce90b10c09e6).[<41>](#Appendix_A_41" \o "Product behavior note 41)

**SESSIONTYPE\_ALTERNATESHELL:** The session is connected by using Terminal Services and is running an alternate shell instead of the standard shell.

**SESSIONTYPE\_REMOTEAPP:** The session is a RAIL (Remote Applications Integrated Locally) session as defined in [[MS-RDPERP]](%5bMS-RDPERP%5d.pdf#Section_832759572d0e4c5288d11b4c998c6bec).

**SESSIONTYPE\_MEDIACENTEREXT:** The session was connected by using a media center extender device. For more information about the media center, see [[MSFT-WINMCE]](https://go.microsoft.com/fwlink/?LinkId=208408).

#### SHADOW\_CONTROL\_REQUEST

The **SHADOW\_CONTROL\_REQUEST** enumeration specifies if a shadow of user input control is being requested.

1. typedef enum
2. {
3. SHADOW\_CONTROL\_REQUEST\_VIEW = 0,
4. SHADOW\_CONTROL\_REQUEST\_TAKECONTROL,
5. SHADOW\_CONTROL\_REQUEST\_Count
6. } SHADOW\_CONTROL\_REQUEST;

**SHADOW\_CONTROL\_REQUEST\_VIEW:** The shadow request is for a view-only session. User input is not being requested.

**SHADOW\_CONTROL\_REQUEST\_TAKECONTROL:** User input control is being requested.

**SHADOW\_CONTROL\_REQUEST\_Count:** Count of enum values.

#### SHADOW\_PERMISSION\_REQUEST

The **SHADOW\_PERMISSION\_REQUEST** enumeration specifies whether user permission is being requested.

1. typedef enum
2. {
3. SHADOW\_PERMISSION\_REQUEST\_SILENT = 0,
4. SHADOW\_PERMISSION\_REQUEST\_REQUESTPERMISSION,
5. SHADOW\_PERMISSION\_REQUEST\_Count
6. } SHADOW\_PERMISSION\_REQUEST;

**SHADOW\_PERMISSION\_REQUEST\_SILENT:** Permission is not requested.

**SHADOW\_PERMISSION\_REQUEST\_REQUESTPERMISSION:** User permission will be requested before the shadow session begins.

**SHADOW\_PERMISSION\_REQUEST\_Count:** Count of enum values.

#### SHADOW\_REQUEST\_RESPONSE

The **SHADOW\_REQUEST\_RESPONSE** enumeration defines the response to a shadow session request.

1. typedef enum
2. {
3. SHADOW\_REQUEST\_RESPONSE\_ALLOW = 0,
4. SHADOW\_REQUEST\_RESPONSE\_DECLINE,
5. SHADOW\_REQUEST\_RESPONSE\_POLICY\_PERMISSION\_REQUIRED,
6. SHADOW\_REQUEST\_RESPONSE\_POLICY\_DISABLED,
7. SHADOW\_REQUEST\_RESPONSE\_POLICY\_VIEW\_ONLY,
8. SHADOW\_REQUEST\_RESPONSE\_POLICY\_VIEW\_ONLY\_PERMISSION\_REQUIRED,
9. SHADOW\_REQUEST\_RESPONSE\_SESSION\_ALREADY\_CONTROLLED
10. } SHADOW\_REQUEST\_RESPONSE;

**SHADOW\_REQUEST\_RESPONSE\_ALLOW:** The user has granted the request for permission to shadow the session.

**SHADOW\_REQUEST\_RESPONSE\_DECLINE:** The user has declined the request for permission to shadow the session.

**SHADOW\_REQUEST\_RESPONSE\_POLICY\_PERMISSION\_REQUIRED:** Permission was not requested, but group policy specifies that permission is required.

**SHADOW\_REQUEST\_RESPONSE\_POLICY\_DISABLED:** Shadowing has been disabled by group policy.

**SHADOW\_REQUEST\_RESPONSE\_POLICY\_VIEW\_ONLY:** A request for control was made, but group policy exclusively allows view-only shadowing.

**SHADOW\_REQUEST\_RESPONSE\_POLICY\_VIEW\_ONLY\_PERMISSION\_REQUIRED:** A request was made to take control without requesting permission, but group policy exclusively allows view-only shadowing and also requires permission.

**SHADOW\_REQUEST\_RESPONSE\_SESSION\_ALREADY\_CONTROLLED:** The session cannot be shadowed because another shadow session is currently controlling the session.

### Structures

#### SESSION\_FILTER

The **SESSION\_FILTER** enumeration specifies the types of filters to apply when retrieving the list of session IDs running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). There is only one type of filter exposed by [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331).

1. typedef enum \_SESSION\_FILTER
2. {
3. SF\_SERVICES\_SESSION\_POPUP
4. } SESSION\_FILTER;

**SF\_SERVICES\_SESSION\_POPUP:** Returns all [**sessions**](#gt_0cd96b80-a737-4f06-bca4-cf9efb449d12) in a logged-on state.

#### PROTOCOLSTATUS\_INFO\_TYPE

The **PROTOCOLSTATUS\_INFO\_TYPE** enumeration specifies the protocol status information requested for a particular session running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998).

1. typedef enum
2. {
3. PROTOCOLSTATUS\_INFO\_BASIC = 0,
4. PROTOCOLSTATUS\_INFO\_EXTENDED = 1,
5. } PROTOCOLSTATUS\_INFO\_TYPE;

**PROTOCOLSTATUS\_INFO\_BASIC:** Returns basic information about the protocol status in a [PROTOCOLSTATUS](#Section_c9066753acbd46789a728fb1b080bd09) structure.

**PROTOCOLSTATUS\_INFO\_EXTENDED:** Returns extended information about the protocol status. Extended information is returned in a [PROTOCOLSTATUSEX](#Section_e32b64ed22e245cab9549e1b52a16c78) structure.

#### QUERY\_SESSION\_DATA\_TYPE

The QUERY\_SESSION\_DATA\_TYPE enumeration specifies the type of session information that can be requested for a particular session running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998).

1. typedef enum
2. {
3. QUERY\_SESSION\_DATA\_MODULE = 0,
4. QUERY\_SESSION\_DATA\_WDCONFIG,
5. QUERY\_SESSION\_DATA\_VIRTUALDATA,
6. QUERY\_SESSION\_DATA\_LICENSE,
7. QUERY\_SESSION\_DATA\_DEVICEID,
8. QUERY\_SESSION\_DATA\_LICENSE\_VALIDATION
9. } QUERY\_SESSION\_DATA\_TYPE;

**QUERY\_SESSION\_DATA\_MODULE:** Retrieves data about protocol-specific binaries loaded for the given [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) session. The type of the data is PBYTE.

**QUERY\_SESSION\_DATA\_WDCONFIG:** Retrieves protocol driver configuration data for the session. The data returned is of type [WDCONFIG](#Section_d1bf099beb544ed9a7230b1062dbc128).

**QUERY\_SESSION\_DATA\_VIRTUALDATA:** Retrieves data about virtual channels for the given Terminal Services session. The data returned is of type PBYTE.

**QUERY\_SESSION\_DATA\_LICENSE:** Retrieves data about the licensing policies associated with a given Terminal Services session.[<42>](#Appendix_A_42" \o "Product behavior note 42)[<43>](#Appendix_A_43" \o "Product behavior note 43)

**QUERY\_SESSION\_DATA\_DEVICEID:** Retrieves the device ID of the client connected to a given Terminal Services session. The data returned is of type PBYTE.[<44>](#Appendix_A_44" \o "Product behavior note 44)

**QUERY\_SESSION\_DATA\_LICENSE\_VALIDATION:** Retrieves the data required to validate the license associated with a given Terminal Services session. The data returned is of type [WINSTATIONVALIDATIONINFORMATION](#Section_5c439d8db58b4b0da2e5467aa05f993b).[<45>](#Appendix_A_45" \o "Product behavior note 45)

#### PSESSIONENUM

PSESSIONENUM is a pointer to a structure containing information about the sessions running on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). It is returned by [RpcGetEnumResult](#Section_1a7d5d1d1ce5448fbb4ac79741982edb).

1. typedef struct \_SESSIONENUM {
2. DWORD Level;
3. [switch\_is(Level)] SessionInfo Data;
4. } SESSIONENUM,
5. \*PSESSIONENUM;

**Level:**  The level of information contained in the **Data** member; the valid values are 1 and 2.

**Data:**  Contains information at a specified level of detail about sessions running on a computer.

##### SessionInfo

The SessionInfo is a union of structures, each structure providing different levels of detail about sessions running on a computer, as specified in sections [2.2.2.4.1.1](#Section_0c4fde3b4fdb45248cae792d0320247a) and [2.2.2.4.1.2](#Section_1b889bfe8d324cbabd92f9d7c09d5f60) respectively.

1. typedef
2. [switch\_type(DWORD)]
3. union \_SessionInfo {
4. [case(1)]
5. SESSIONENUM\_LEVEL1 SessionEnum\_Level1;
6. [case(2)]
7. SESSIONENUM\_LEVEL2 SessionEnum\_Level2;
8. } SessionInfo,
9. \*PSessionInfo;

**SessionEnum\_Level1:**  A SESSIONENUM\_LEVEL1 structure containing a level of information about sessions running on a computer.

**SessionEnum\_Level2:**  A SESSIONENUM\_LEVEL2 structure containing a level of information about sessions running on a computer.

###### SESSIONENUM\_LEVEL1

The SESSIONENUM\_LEVEL1 structure contains basic information about sessions running on a computer.

1. typedef struct \_SESSIONENUM\_LEVEL1 {
2. LONG SessionId;
3. LONG State;
4. WCHAR Name[33];
5. } SESSIONENUM\_LEVEL1,
6. \*PSESSIONENUM\_LEVEL1;

**SessionId:**  An identifier assigned by the operating system to the session contained in this structure.

**State:**  The state of the session, as specified in section [3.3.4.1.8](#Section_58b06e1dbfe44a2f90a19c9129923e99).

**Name:**  A string that contains the name of the session assigned by [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) followed by the terminating NULL character.

###### SESSIONENUM\_LEVEL2

The SESSIONENUM\_LEVEL2 structure contains information about sessions running on a computer that is more detailed than the information contained in [SESSIONENUM\_LEVEL1](#Section_0c4fde3b4fdb45248cae792d0320247a).

1. typedef struct \_SESSIONENUM\_LEVEL2 {
2. LONG SessionId;
3. LONG State;
4. WCHAR Name[33];
5. ULONG Source;
6. BOOL bFullDesktop;
7. GUID SessionType;
8. } SESSIONENUM\_LEVEL2,
9. \*PSESSIONENUM\_LEVEL2;

**SessionId:**  An identifier assigned by the operating system to the session contained in this structure.

**State:**  The state of the session, as specified in section [3.3.4.1.8](#Section_58b06e1dbfe44a2f90a19c9129923e99).

**Name:**  A string that contains the name of the session followed by the terminating NULL character.

**Source:**  The parameter is always set to zero.

**bFullDesktop:**  The parameter is always set to TRUE.

**SessionType:**  Describes the type of the session.[<46>](#Appendix_A_46" \o "Product behavior note 46)

###### SESSIONENUM\_LEVEL3

The SESSIONENUM\_LEVEL3 structure contains information about sessions running on a computer that is more detailed than the information contained in [SESSIONENUM\_LEVEL1](#Section_0c4fde3b4fdb45248cae792d0320247a) and [SESSIONENUM\_LEVEL2](#Section_1b889bfe8d324cbabd92f9d7c09d5f60).

1. typedef struct \_SESSIONENUM\_LEVEL3 {
2. LONG SessionId;
3. LONG State;
4. WCHAR Name[33];
5. ULONG Source;
6. BOOL bFullDesktop;
7. GUID SessionType;
8. ULONG ProtoDataSize;
9. [size\_is(ProtoDataSize)] UCHAR\* pProtocolData;
10. } SESSIONENUM\_LEVEL3,
11. \*PSESSIONENUM\_LEVEL3;

**SessionId:**  An identifier assigned by the operating system to the session contained in this structure.

**State:**  The state of the session, as specified in section [3.3.4.1.8](#Section_58b06e1dbfe44a2f90a19c9129923e99).

**Name:**  A string that contains the name of the session followed by the terminating NULL character.

**Source:**  The parameter is always set to zero.

bFullDesktop:  The parameter is always set to TRUE.

SessionType:  The parameter is always set to zero.

ProtoDataSize:  Size of data, in bytes, contained in the pProtocolData member.

pProtocolData:  Data about the protocol status between the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) client and server. This data will be of type [PROTOCOLSTATUSEX](#Section_e32b64ed22e245cab9549e1b52a16c78).

#### PSESSIONENUM\_EX

The PSESSIONENUM\_EX is a pointer to a structure containing information about the sessions running on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). It is returned by [RpcGetEnumResultEx](#Section_cb2be08441dd44ec81dc52d371a28554).

1. typedef struct \_SESSIONENUM\_EX {
2. DWORD Level;
3. [switch\_is(Level)] SessionInfo\_Ex Data;
4. } SESSIONENUM\_EX,
5. \*PSESSIONENUM\_EX;

**Level:**  The level of information contained in **Data**; the valid values are 1, 2, and 3.

| Value | Meaning |
| --- | --- |
| 1 | The union [SessionInfo\_Ex](#Section_1cae32cd3bd3436db9501f43c2b40863) has the [SessionEnum\_Level1](#Section_0c4fde3b4fdb45248cae792d0320247a) structure. |
| 2 | The union SessionInfo\_Ex has the [SessionEnum\_Level2](#Section_1b889bfe8d324cbabd92f9d7c09d5f60) structure. |
| 3 | The union SessionInfo\_Ex has the [SessionEnum\_Level3](#Section_d1662f979e3a4fbbbdd52050df69e902) structure. |

**Data:**  Contains information at a specified level of detail about sessions running on a computer. This parameter is of type SessionInfo\_Ex. If Level is set to 1, the union SessionInfo\_Ex has the SessionEnum\_Level1 structure. If Level is set to 2, the union SessionInfo\_Ex has the SessionEnum\_Level2 structure. If Level is set to 3, the union SessionInfo\_Ex has the SessionEnum\_Level3 structure.

##### SessionInfo\_Ex

The SessionInfo\_Ex is a union of structures, each structure providing different levels of detail about sessions running on a computer, as specified in sections [2.2.2.4.1.1](#Section_0c4fde3b4fdb45248cae792d0320247a), [2.2.2.4.1.2](#Section_1b889bfe8d324cbabd92f9d7c09d5f60), and [2.2.2.4.1.3](#Section_d1662f979e3a4fbbbdd52050df69e902) respectively.

1. typedef
2. [switch\_type(DWORD)]
3. union \_SessionInfo\_Ex {
4. [case(1)]
5. SESSIONENUM\_LEVEL1 SessionEnum\_Level1;
6. [case(2)]
7. SESSIONENUM\_LEVEL2 SessionEnum\_Level2;
8. [case(3)]
9. SESSIONENUM\_LEVEL3 SessionEnum\_Level3;
10. } SessionInfo\_Ex,
11. \*PSessionInfo\_Ex;

**SessionEnum\_Level1:**  A SESSIONENUM\_LEVEL1 structure containing a level of information about sessions running on a computer.

**SessionEnum\_Level2:**  A SESSIONENUM\_LEVEL2 structure containing a level of information about sessions running on a computer.

**SessionEnum\_Level3:**  A SESSIONENUM\_LEVEL3 structure containing a level of information about sessions running on a computer.

#### PEXECENVDATA

PEXECENVDATA is a pointer to a structure containing information about the [**sessions**](#gt_0cd96b80-a737-4f06-bca4-cf9efb449d12) running on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) and the sessions running on virtual machines hosted on the server.[<47>](#Appendix_A_47" \o "Product behavior note 47) It is returned by [RpcGetAllSessions](#Section_68e824e0d78649878ccbc34503fffd3f).

1. typedef struct \_EXECENVDATA {
2. DWORD Level;
3. [switch\_is(Level)] ExecEnvData Data;
4. } EXECENVDATA,
5. \*PEXECENVDATA;

**Level:**  The level of information contained in **Data**; the valid values are 1 and 2.

| Value | Meaning |
| --- | --- |
| 1 | The union [ExecEnvData](#Section_9846fe50582c4f69a070677049ebc1ae) has the [EXECENVDATA\_LEVEL1](#Section_004305a642654ff5bb8a58fd0f7cf2d5) structure. |
| 2 | The union ExecEnvData has the [EXECENVDATA\_LEVEL2](#Section_e85daed7ed774c8aafc05596c8d41c9c) structure. |

**Data:**  Contains information at a specified level of detail about sessions running on a computer. This is of type ExecEnvData.

##### ExecEnvData

The ExecEnvData is a union of structures, each structure providing different levels of detail about sessions running on a computer and sessions running on virtual machines hosted on the server,[<48>](#Appendix_A_48" \o "Product behavior note 48) as specified in sections [2.2.2.6.1.1](#Section_004305a642654ff5bb8a58fd0f7cf2d5) and [2.2.2.6.1.2](#Section_e85daed7ed774c8aafc05596c8d41c9c) respectively.

1. typedef
2. [switch\_type(DWORD)]
3. union \_ExecEnvData {
4. [case(1)]
5. EXECENVDATA\_LEVEL1 ExecEnvEnum\_Level1;
6. [case(2)]
7. EXECENVDATA\_LEVEL2 ExecEnvEnum\_Level2;
8. } ExecEnvData,
9. \*PExecEnvData;

**ExecEnvEnum\_Level1:**  An EXECENVDATA\_LEVEL1 structure containing a level of information about sessions running on a computer and virtual machines hosted on the computer.[<49>](#Appendix_A_49" \o "Product behavior note 49)

**ExecEnvEnum\_Level2:**  An EXECENVDATA\_LEVEL2 structure containing a level of information about sessions running on a computer and virtual machines hosted on the computer.[<50>](#Appendix_A_50" \o "Product behavior note 50)

###### EXECENVDATA\_LEVEL1

The EXECENVDATA\_LEVEL1 structure contains basic information about sessions running on a computer.

1. typedef struct \_EXECENVDATA\_LEVEL1 {
2. LONG ExecEnvId;
3. LONG State;
4. WCHAR SessionName[33];
5. } EXECENVDATA\_LEVEL1,
6. \*PEXECENVDATA\_LEVEL1;

**ExecEnvId:**  An identifier assigned to the session contained in this structure by the component that aggregates the sessions on the server and sessions within virtual machines hosted on the server.[<51>](#Appendix_A_51" \o "Product behavior note 51)

**State:**  The state of the session, as specified in section [3.3.4.1.8](#Section_58b06e1dbfe44a2f90a19c9129923e99).

**SessionName:**  A string that contains the name of the session assigned by [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) followed by the terminating NULL character.

###### EXECENVDATA\_LEVEL2

The EXECENVDATA\_LEVEL2 structure contains information about sessions running on a computer that is more detailed than the information contained in [EXECENVDATA\_LEVEL1](#Section_004305a642654ff5bb8a58fd0f7cf2d5).

1. typedef struct \_EXECENVDATA\_LEVEL2 {
2. LONG ExecEnvId;
3. LONG State;
4. WCHAR SessionName[33];
5. LONG AbsSessionId;
6. WCHAR HostName[33];
7. WCHAR UserName[33];
8. WCHAR DomainName[33];
9. WCHAR FarmName[33];
10. } EXECENVDATA\_LEVEL2,
11. \*PEXECENVDATA\_LEVEL2;

**ExecEnvId:**  An identifier assigned to the session contained in this structure by the component that aggregates the sessions on the server and sessions within virtual machines hosted on the server.[<52>](#Appendix_A_52" \o "Product behavior note 52)

**State:**  The state of the session, as specified in section [3.3.4.1.8](#Section_58b06e1dbfe44a2f90a19c9129923e99).

**SessionName:**  A string that contains the name of the session followed by the terminating NULL character.

**AbsSessionId:**  An identifier assigned by the operating system running in the virtual machine to the session contained in this structure. If the session contained in this structure is not running under the virtual machine, the value of **AbsSessionId** is same as **ExecEnvId**.

**HostName:**  A string that contains the name of the machine that hosts the session contained in this structure followed by the terminating NULL character.

**UserName:**  A string that contains the name of the user logged onto the session followed by the terminating NULL character.

**DomainName:**  A string that contains the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) of the user logged onto the session followed by the terminating NULL character.

**FarmName:**  A string that contains the farm name associated with the session followed by the terminating NULL character.

#### PEXECENVDATAEX

**PEXECENVDATAEX** is a pointer to a structure containing information about the sessions running on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) and the sessions running on virtual machines hosted on the server.[<53>](#Appendix_A_53" \o "Product behavior note 53) It is returned by [RpcGetAllSessionsEx](#Section_3f72497925cd41d89829244f4425f9c6).

1. typedef struct \_EXECENVDATAEX {
2. DWORD Level;
3. [switch\_is(Level)] ExecEnvDataEx Data;
4. } EXECENVDATAEX,
5. \*PEXECENVDATAEX;

**Level:**  The level of information contained in the **Data** member; the only valid value is 1.

| Value | Meaning |
| --- | --- |
| 1 | The union [ExecEnvDataEx](#Section_d77326bf8ed74e1d9429965c2cd97f37) has the [EXECENVDATAEX\_LEVEL1](#Section_b4295b0ec64140abbbdbea44323b0804) structure. |

**Data:**  Contains information at a specified level of detail about sessions running on a computer. This is of type ExecEnvDataEx.

##### ExecEnvDataEx

**ExecEnvDataEx** is a union of structures that provides information about sessions running on a computer and sessions running on virtual machines hosted on the server,[<54>](#Appendix_A_54" \o "Product behavior note 54) as specified in section [2.2.2.7.1.1](#Section_b4295b0ec64140abbbdbea44323b0804).

1. typedef
2. [switch\_type(DWORD)]
3. union \_ExecEnvDataEx {
4. [case(1)]
5. EXECENVDATAEX\_LEVEL1 ExecEnvEnum\_Level1;
6. } ExecEnvDataEx,
7. \*PExecEnvDataEx;

**ExecEnvEnum\_Level1:**  An EXECENVDATAEX\_LEVEL1 structure that contains information about sessions running on a computer and virtual machines hosted on the computer.[<55>](#Appendix_A_55" \o "Product behavior note 55)

###### EXECENVDATAEX\_LEVEL1

The **EXECENVDATAEX\_LEVEL1** structure contains information about sessions running on a computer.

1. typedef struct \_EXECENVDATAEX\_LEVEL1 {
2. LONG ExecEnvId;
3. LONG State;
4. LONG AbsSessionId;
5. [string, max\_is (256)] LPWSTR pszSessionName;
6. [string, max\_is (256)] LPWSTR pszHostName;
7. [string, max\_is (256)] LPWSTR pszUserName;
8. [string, max\_is (256)] LPWSTR pszDomainName;
9. [string, max\_is (256)] LPWSTR pszFarmName;
10. } EXECENVDATAEX\_LEVEL1,
11. \*PEXECENVDATAEX\_LEVEL1;

**ExecEnvId:**  An identifier assigned to the session contained in this structure by the component that aggregates the sessions on the server and sessions within virtual machines hosted on the server.

**State:**  The state of the session, as specified in section [3.3.4.1.8](#Section_58b06e1dbfe44a2f90a19c9129923e99).

**AbsSessionId:**  An identifier assigned by the operating system running in the virtual machine to the session contained in this structure. If the session contained in this structure is not running under the virtual machine, the value of **AbsSessionId** is the same as the value of the **ExecEnvId** member.

**pszSessionName:**  A string that contains the name of the session followed by the terminating NULL character.

**pszHostName:**  A string that contains the name of the machine that hosts the session contained in this structure, followed by the terminating NULL character.

**pszUserName:**  A string that contains the name of the user logged onto the session followed by the terminating NULL character.

**pszDomainName:**  A string that contains the [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) of the user logged onto the session followed by the terminating NULL character.

**pszFarmName:**  A string that contains the farm name associated with the session followed by the terminating NULL character.

#### PLSMSESSIONINFORMATION

PLSMSESSIONINFORMATION is a pointer to a LSMSESSIONINFORMATION structure containing information about a session running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998).

1. typedef struct \_LSMSessionInformation {
2. [string] WCHAR\* pszUserName;
3. [string] WCHAR\* pszDomain;
4. [string] WCHAR\* pszTerminalName;
5. LONG SessionState;
6. BOOL DesktopLocked;
7. hyper ConnectTime;
8. hyper DisconnectTime;
9. hyper LogonTime;
10. } LSMSESSIONINFORMATION,
11. \*PLSMSESSIONINFORMATION;

**pszUserName:**  The name of the user logged on to the session.

**pszDomain:**  The [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) to which the currently logged-on user belongs.

**pszTerminalName:**  The name of the terminal associated with the specific session.

**SessionState:**  The state of the session, as described in section [3.3.4.1.8](#Section_58b06e1dbfe44a2f90a19c9129923e99).

**DesktopLocked:**  Set to TRUE if the session is currently locked; FALSE otherwise.

**ConnectTime:**  The time of the most recent connection to the session.

Time is measured as the number of 100-nanosecond intervals since January 1, 1601 (UTC).

**DisconnectTime:**  The time of the most recent disconnection from the session.

Time is measured as the number of 100-nanosecond intervals since January 1, 1601 (UTC).

**LogonTime:**  The time of the most recent logon to the session.

Time is measured as the number of 100-nanosecond intervals since January 1, 1601 (UTC).

#### PLSMSESSIONINFORMATION\_EX

The PLSMSESSIONINFORMATION\_EX is a pointer to a LSMSESSIONINFORMATION\_EX structure containing information about a session running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) and the level of detail of the information provided.

1. typedef struct \_LSMSESSIONINFORMATION\_EX {
2. DWORD Level;
3. [switch\_is(Level)] LSM\_SESSIONINFO\_EX Data;
4. } LSMSESSIONINFORMATION\_EX,
5. \*PLSMSESSIONINFORMATION\_EX;

**Level:**  The level of detail provided about the session. This field MUST be set to 1.

**Data:**  Information about the session. This is of type [LSM\_SESSIONINFO\_EX](#Section_8998a977b6a34bf8863a6cb9f54d9d14).

#### LSM\_SESSIONINFO\_EX

The LSM\_SESSIONINFO\_EX is a union of structures, each member containing a different level of information about a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) session.

1. typedef
2. [switch\_type(DWORD)]
3. union \_LSM\_SESSIONINFO\_EX {
4. [case(1)]
5. LSM\_SESSIONINFO\_EX\_LEVEL1 LSM\_SessionInfo\_Level1;
6. } LSM\_SESSIONINFO\_EX,
7. \*PLSM\_SESSIONINFO\_EX;

**LSM\_SessionInfo\_Level1:**  The only supported member of the union. It contains session information of level 1. It is of type [LSM\_SESSIONINFO\_EX\_LEVEL1](#Section_2789ef970c7c4ec59f2cd375de4941a0).

#### LSM\_SESSIONINFO\_EX\_LEVEL1

The LSM\_SESSIONINFO\_EX\_LEVEL1 is a structure containing information about a session running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998).

1. typedef struct \_LSM\_SESSIONINFO\_EX\_LEVEL1 {
2. LONG SessionState;
3. LONG SessionFlags;
4. WCHAR SessionName[33];
5. WCHAR DomainName[18];
6. WCHAR UserName[21];
7. hyper ConnectTime;
8. hyper DisconnectTime;
9. hyper LogonTime;
10. hyper LastInputTime;
11. ULONG ProtocolDataSize;
12. [size\_is(ProtocolDataSize)] PBYTE ProtocolData;
13. } LSM\_SESSIONINFO\_EX\_LEVEL1,
14. \*PLSM\_SESSIONINFO\_EX\_LEVEL1;

**SessionState:**  The state of the session, as described in section [3.3.4.1.8](#Section_58b06e1dbfe44a2f90a19c9129923e99).

**SessionFlags:**  The state of the session. The **SessionFlags** member MUST be one of the values shown in the following table.

| Value | Meaning |
| --- | --- |
| WTS\_SESSIONSTATE\_UNKNOWN  0xFFFFFFFF | Unknown session state |
| WTS\_SESSIONSTATE\_LOCK  0x00000000 | Session is locked |
| WTS\_SESSIONSTATE\_UNLOCK  0x00000001 | Session is unlocked |

**SessionName:**  The name of the terminal associated with the specific session.

**DomainName:**  The [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) to which the currently logged-on user belongs.

**UserName:**  The name of the user logged on to the session.

**ConnectTime:**  The time of the most recent connection to the session.

Time is measured as the number of 100-nanosecond intervals since January 1, 1601 (UTC).

**DisconnectTime:**  The time of the most recent disconnection from the session.

Time is measured as the number of 100-nanosecond intervals since January 1, 1601 (UTC).

**LogonTime:**  The time of the most recent logon to the session.

Time is measured as the number of 100-nanosecond intervals since January 1, 1601 (UTC).

**LastInputTime:**  The time the session last received input. This is an indicator of how long a session has been idle.

Time is measured as the number of 100-nanosecond intervals since January 1, 1601 (UTC).

**ProtocolDataSize:**  Size of data, in bytes, contained in **ProtocolData**.

**ProtocolData:**  Data about the protocol status between the terminal server client and server. This data is of type [PROTOCOLSTATUSEX](#Section_e32b64ed22e245cab9549e1b52a16c78).

#### PLISTENERENUM

PLISTENERENUM contains information about one [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) [**listener**](#gt_1e5aa0c2-4617-4d7a-969e-a7091c415c8d) and the level of detail of the information provided.

1. typedef struct \_LISTENERENUM {
2. DWORD Level;
3. [switch\_is(Level)] ListenerInfo Data;
4. } LISTENERENUM,
5. \*PLISTENERENUM;

**Level:**  The level of detail provided about the listener. The only supported value is 1.

**Data:**  Information about the listener. This is of the type [ListenerInfo](#Section_9efbd65053fe4cf68b4f48bc3f0adc58).

##### ListenerInfo

ListenerInfo is a union of structures, each member containing a different level of information about a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) [**listener**](#gt_1e5aa0c2-4617-4d7a-969e-a7091c415c8d).

1. typedef
2. [switch\_type(DWORD)]
3. union \_ListenerInfo {
4. [case(1)]
5. LISTENERENUM\_LEVEL1 ListenerEnum\_Level1;
6. [default]  ;
7. } ListenerInfo,
8. \*PListenerInfo;

**ListenerEnum\_Level1:**  The only supported member of the union. It contains listener information of level 1. It is of the type [LISTENERENUM\_LEVEL1](#Section_810c8c1548e342229c5058cd21c729fb).

###### LISTENERENUM\_LEVEL1

LISTENERENUM\_LEVEL1 is a structure containing information of level 1 detail about a [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) [**listener**](#gt_1e5aa0c2-4617-4d7a-969e-a7091c415c8d).

1. typedef struct \_LISTENERENUM\_LEVEL1 {
2. LONG Id;
3. BOOL bListening;
4. WCHAR Name[33];
5. } LISTENERENUM\_LEVEL1,
6. \*PLISTENERENUM\_LEVEL1;

**Id:**  The identifier associated with the listener.

**bListening:**   Set to TRUE if the listener is listening for incoming connections; FALSE otherwise.

**Name:**  A string that contains the name of the listener followed by the terminating NULL character.

#### LOGONID

LOGONID is a macro defined to be the structure **SESSIONID**. This type represents information about the session or [**WinStation**](#gt_8cc89bb9-be94-4672-8a5d-9947b91bc952) identified by the identifier **SessionId**. For more information, see the macro definition in section [6.5](#Section_ce70794f213843e8bf6c2c147887d6a2).

1. typedef struct \_SESSIONID {
2. union {
3. ULONG SessionId;
4. ULONG LogonId;
5. } \_SessionId\_LogonId\_union;
6. WINSTATIONNAME WinStationName;
7. WINSTATIONSTATECLASS State;
8. } SESSIONID,
9. \*PSESSIONID;

**SessionId:**  In a union with LogonId. It represents WinStation or session identifier numbered 0 through 65535 for [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) sessions. A number of 65536 or greater indicates that the WinStation is a listening WinStation.

**LogonId:**  In a union with **SessionId**. It is used internally only, within Terminal Services code.

**WinStationName:**  The name of the WinStation represented by this structure. See section [2.2.1.5](#Section_b34d2909b0cd407e934ca2c7cb96ca26) for more information on the type WINSTATIONNAME.

**State:**  The current state of the WinStation. See section [2.2.1.9](#Section_b619d8aa4e8a4716954ea4d596ed37f8) for more information on the type WINSTATIONSTATECLASS.

#### TS\_PROCESS\_INFORMATION\_NT4

The TS\_PROCESS\_INFORMATION\_NT4 structure is returned by [RpcWinStationEnumerateProcesses](#Section_3559bc198ca34bcf842ecf25f592dc1a).

1. typedef struct \_TS\_PROCESS\_INFORMATION\_NT4 {
2. ULONG MagicNumber;
3. ULONG LogonId;
4. PVOID ProcessSid;
5. ULONG Pad;
6. } TS\_PROCESS\_INFORMATION\_NT4,
7. \*PTS\_PROCESS\_INFORMATION\_NT4;

**MagicNumber:**  MUST be set to TS\_PROCESS\_INFO\_MAGIC\_NT4 (0x23495452).

**LogonId:**  The session ID of the process.

**ProcessSid:**  The security identifier (SID), as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.2, of the owner of the process.

**Pad:**  MUST be set to 0.

#### TS\_ALL\_PROCESSES\_INFO

The **TS\_ALL\_PROCESSES\_INFO** structure contains data on all the processes on the system accessible to the user who issued the call.

1. typedef struct \_TS\_ALL\_PROCESSES\_INFO {
2. PTS\_SYS\_PROCESS\_INFORMATION pTsProcessInfo;
3. DWORD SizeOfSid;
4. #ifdef \_\_midl
5. [size\_is(SizeOfSid)] PBYTE pSid;
6. #else
7. PBYTE pSid;
8. #endif
9. } TS\_ALL\_PROCESSES\_INFO,
10. \*PTS\_ALL\_PROCESSES\_INFO;

**pTsProcessInfo:**  Pointer to the process information [TS\_SYS\_PROCESS\_INFORMATION](#Section_96702c7d6f5f49658f295fdbea73278e).

**SizeOfSid:**  Size of **pSid**, in bytes.

**pSid:**  The security identifier (SID), as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2), of the owner of the process.

##### TS\_SYS\_PROCESS\_INFORMATION

The TS\_SYS\_PROCESS\_INFORMATION structure contains information about a process running on a system.

1. typedef struct \_TS\_SYS\_PROCESS\_INFORMATION {
2. ULONG NextEntryOffset;
3. ULONG NumberOfThreads;
4. LARGE\_INTEGER SpareLi1;
5. LARGE\_INTEGER SpareLi2;
6. LARGE\_INTEGER SpareLi3;
7. LARGE\_INTEGER CreateTime;
8. LARGE\_INTEGER UserTime;
9. LARGE\_INTEGER KernelTime;
10. TS\_UNICODE\_STRING ImageName;
11. LONG BasePriority;
12. DWORD UniqueProcessId;
13. DWORD InheritedFromUniqueProcessId;
14. ULONG HandleCount;
15. ULONG SessionId;
16. ULONG SpareUl3;
17. SIZE\_T PeakVirtualSize;
18. SIZE\_T VirtualSize;
19. ULONG PageFaultCount;
20. ULONG PeakWorkingSetSize;
21. ULONG WorkingSetSize;
22. SIZE\_T QuotaPeakPagedPoolUsage;
23. SIZE\_T QuotaPagedPoolUsage;
24. SIZE\_T QuotaPeakNonPagedPoolUsage;
25. SIZE\_T QuotaNonPagedPoolUsage;
26. SIZE\_T PagefileUsage;
27. SIZE\_T PeakPagefileUsage;
28. SIZE\_T PrivatePageCount;
29. } TS\_SYS\_PROCESS\_INFORMATION,
30. \*PTS\_SYS\_PROCESS\_INFORMATION;

**NextEntryOffset:**  Offset to the start of data for the next process.

**NumberOfThreads:**  Number of threads in the process.

**SpareLi1:**  Reserved.

**SpareLi2:**  Reserved.

**SpareLi3:**  Reserved.

**CreateTime:**  Creation time of the process. Time is measured as the number of 100-nanosecond intervals since January 1, 1601 (UTC).

**UserTime:**  Amount of time in milliseconds the process has spent running in user mode.

**KernelTime:**  Amount of time in milliseconds the process has spent running in kernel mode.

**ImageName:**  String containing the process's image name.

**BasePriority:**  Base priority of the process.

**UniqueProcessId:**  Process's unique process ID.

**InheritedFromUniqueProcessId:**  Parent process's unique process ID.

**HandleCount:**  Current number of handles open in the process.

**SessionId:**  Session identifier of the process session.

**SpareUl3:**  Reserved.

**PeakVirtualSize:**  Peak size of virtual memory, in bytes, used by the process.

**VirtualSize:**  Current size of virtual memory, in bytes, used by the process.

**PageFaultCount:**  Number of page faults in the process.

**PeakWorkingSetSize:**  Peak size of the working set in kilobytes of the process.

**WorkingSetSize:**  Current size, in bytes, of the working set of the process.

**QuotaPeakPagedPoolUsage:**  Peak quota charged to the process for paged pool usage.

**QuotaPagedPoolUsage:**  Current quota charged to the process for paged pool usage.

**QuotaPeakNonPagedPoolUsage:**  Peak quota charged to the process for nonpaged pool usage.

**QuotaNonPagedPoolUsage:**  Current quota charged to the process for nonpaged pool usage.

**PagefileUsage:**  Amount of bytes of page file storage in use by the process.

**PeakPagefileUsage:**  Peak amount of bytes of page file storage in use by the process.

**PrivatePageCount:**  Current number of memory pages allocated by the process.

###### TS\_UNICODE\_STRING

The **TS\_UNICODE\_STRING** structure contains a [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string.

1. typedef struct \_TS\_UNICODE\_STRING {
2. USHORT Length;
3. USHORT MaximumLength;
4. #ifdef \_\_midl
5. [size\_is(MaximumLength),length\_is(Length)]PWSTR Buffer;
6. #else
7. PWSTR Buffer;
8. #endif
9. } TS\_UNICODE\_STRING;

**Length:**  The actual length of the string currently stored in the **Buffer** member, in bytes.

**MaximumLength:**  The maximum length of the string that can be stored in **Buffer**, in bytes.

**Buffer:**  A wide character string that MUST NOT be followed by the terminating NULL character.

#### TS\_ALL\_PROCESSES\_INFO\_NT6

The **TS\_ALL\_PROCESSES\_INFO\_NT6** structure contains data on all the processes on the system that are accessible using the user's credentials.

1. typedef struct \_TS\_ALL\_PROCESSES\_INFO\_NT6 {
2. PTS\_SYS\_PROCESS\_INFORMATION\_NT6 pTsProcessInfo;
3. DWORD SizeOfSid;
4. #ifdef \_\_midl
5. [size\_is(SizeOfSid)] PBYTE pSid;
6. #else
7. PBYTE pSid;
8. #endif
9. } TS\_ALL\_PROCESSES\_INFO\_NT6, \*PTS\_ALL\_PROCESSES\_INFO\_NT6;

**pTsProcessInfo:**  Pointer to the process information.

**SizeOfSid:**  Size, in bytes, of the security identifier (SID) structure pointed to by **pSid**.

**pSid:**  Security identifier (SID), as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.2, of the process.

##### TS\_SYS\_PROCESS\_INFORMATION\_NT6

The TS\_SYS\_PROCESS\_INFORMATION\_NT6 structure contains information about a process running on a system.

1. typedef struct \_TS\_SYS\_PROCESS\_INFORMATION\_NT6 {
2. ULONG NextEntryOffset;
3. ULONG NumberOfThreads;
4. LARGE\_INTEGER SpareLi1;
5. LARGE\_INTEGER SpareLi2;
6. LARGE\_INTEGER SpareLi3;
7. LARGE\_INTEGER CreateTime;
8. LARGE\_INTEGER UserTime;
9. LARGE\_INTEGER KernelTime;
10. NT6\_TS\_UNICODE\_STRING ImageName;
11. LONG BasePriority;
12. DWORD UniqueProcessId;
13. DWORD InheritedFromUniqueProcessId;
14. ULONG HandleCount;
15. ULONG SessionId;
16. ULONG SpareUl3;
17. SIZE\_T PeakVirtualSize;
18. SIZE\_T VirtualSize;
19. ULONG PageFaultCount;
20. ULONG PeakWorkingSetSize;
21. ULONG WorkingSetSize;
22. SIZE\_T QuotaPeakPagedPoolUsage;
23. SIZE\_T QuotaPagedPoolUsage;
24. SIZE\_T QuotaPeakNonPagedPoolUsage;
25. SIZE\_T QuotaNonPagedPoolUsage;
26. SIZE\_T PagefileUsage;
27. SIZE\_T PeakPagefileUsage;
28. SIZE\_T PrivatePageCount;
29. } TS\_SYS\_PROCESS\_INFORMATION\_NT6,
30. \*PTS\_SYS\_PROCESS\_INFORMATION\_NT6;

**NextEntryOffset:**  Offset to the start of data for the next process.

**NumberOfThreads:**  Number of threads in the process.

**SpareLi1:**  Reserved.

**SpareLi2:**  Reserved.

**SpareLi3:**  Reserved.

**CreateTime:**  Creation time of the process. Time is measured as the number of 100-nanosecond intervals since January 1, 1601 (UTC).

**UserTime:**  Amount of time in milliseconds the process has spent running in user mode.

**KernelTime:**  Amount of time in milliseconds the process has spent running in kernel mode.

**ImageName:**  String containing the process's image name.

**BasePriority:**  Base priority of the process, which is the starting priority for threads created within the associated process.

**UniqueProcessId:**  Process's unique process ID.

**InheritedFromUniqueProcessId:**  Parent process's unique process ID.

**HandleCount:**  Current number of handles open in the process.

**SessionId:**  Session identifier of the process session.

**SpareUl3:**  Reserved.

**PeakVirtualSize:**  Peak size, in bytes, of the virtual memory used by the process.

**VirtualSize:**  Current size, in bytes, of virtual memory used by the process.

**PageFaultCount:**  Number of page faults in the process.

**PeakWorkingSetSize:**  Peak size, in kilobytes, of the working set of the process.

**WorkingSetSize:**  Current size, in bytes, of the working set of the process.

**QuotaPeakPagedPoolUsage:**  Peak quota charged to the process for paged pool usage.

**QuotaPagedPoolUsage:**  Current quota charged to the process for paged pool usage.

**QuotaPeakNonPagedPoolUsage:**  Peak quota charged to the process for nonpaged pool usage.

**QuotaNonPagedPoolUsage:**  Current quota charged to the process for nonpaged pool usage.

**PagefileUsage:**  Number of bytes of page file storage in use by the process.

**PeakPagefileUsage:**  Peak number of bytes of page file storage in use by the process.

**PrivatePageCount:**  Current number of memory pages allocated by the process.

###### NT6\_TS\_UNICODE\_STRING

The **NT6\_TS\_UNICODE\_STRING** structure contains a [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string.

1. typedef struct \_NT6\_TS\_UNICODE\_STRING {
2. USHORT Length;
3. USHORT MaximumLength;
4. #ifdef \_\_midl
5. [size\_is(MaximumLength / 2),length\_is(Length / 2)]PWSTR Buffer;
6. #else
7. PWSTR Buffer;
8. #endif
9. } NT6\_TS\_UNICODE\_STRING;

**Length:**  The actual length of the string currently stored in **Buffer**, in bytes.

**MaximumLength:**  The maximum length of the string that could be stored in **Buffer**, in bytes.

**Buffer:**  A wide character string that MUST NOT be followed by the terminating NULL character.

##### SYSTEM\_THREAD\_INFORMATION

The **SYSTEM\_THREAD\_INFORMATION** structure contains information about a thread running on a system.

1. typedef struct \_SYSTEM\_THREAD\_INFORMATION {
2. LARGE\_INTEGER KernelTime;
3. LARGE\_INTEGER UserTime;
4. LARGE\_INTEGER CreateTime;
5. ULONG WaitTime;
6. PVOID StartAddress;
7. CLIENT\_ID ClientId;
8. LONG Priority;
9. LONG BasePriority;
10. ULONG ContextSwitches;
11. ULONG ThreadState;
12. ULONG WaitReason;
13. } SYSTEM\_THREAD\_INFORMATION,
14. \*PSYSTEM\_THREAD\_INFORMATION;

**KernelTime:**  Number of 100-nanosecond intervals spent executing kernel code.

**UserTime:**  Number of 100-nanosecond intervals spent executing user code.

**CreateTime:**  System time when the thread was created.

**WaitTime:**  Time spent in ready queue or waiting (depending on the thread state).

**StartAddress:**  Start address of the thread.

**ClientId:**  ID of the thread and the process owning the thread.

**Priority:**  Dynamic thread priority.

**BasePriority:**  Base thread priority.

**ContextSwitches:**  Total context switches.

**ThreadState:**  Current thread state.

**WaitReason:**  The reason the thread is waiting.

###### CLIENT\_ID

The **CLIENT\_ID** structure contains identifiers of a process and a thread.

1. typedef struct \_CLIENT\_ID {
2. HANDLE UniqueProcess;
3. HANDLE UniqueThread;
4. } CLIENT\_ID;

**UniqueProcess:**  Unique process identifier.

**UniqueThread:**  Unique thread identifier.

#### TS\_COUNTER

A [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) performance counter structure used to represent a single performance counter.

1. typedef struct \_TS\_COUNTER {
2. TS\_COUNTER\_HEADER counterHead;
3. DWORD dwValue;
4. LARGE\_INTEGER startTime;
5. } TS\_COUNTER,
6. \*PTS\_COUNTER;

**counterHead:**  A header identifying the counter.

**dwValue:**  The value of the counter. This indicates different things based on the counter.

**startTime:**  Always set to zero because time stamps are not supported.

##### TS\_COUNTER\_HEADER

The **TS\_COUNTER\_HEADER** is the header of the [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) performance counter structure providing general information on the counter.

1. typedef struct \_TS\_COUNTER\_HEADER {
2. DWORD dwCounterID;
3. boolean bResult;
4. } TS\_COUNTER\_HEADER, \*PTS\_COUNTER\_HEADER;

**dwCounterID:**  The identifier of the counter. Set by the caller of [RpcWinStationGetTermSrvCountersValue](#Section_46f107c4f0904802b6d8c510cfcb9a93) to indicate the counter on which to retrieve data. This will be set to zero by RpcWinStationGetTermSrvCountersValue if the **dwCounterId** isn't recognized.

The following values for dwCounterId are supported.

| Value | Meaning |
| --- | --- |
| TERMSRV\_TOTAL\_SESSIONS  0x01 | **Total number of sessions**: Value will indicate the total number of reconnections to the server since startup. |
| TERMSRV\_DISC\_SESSIONS  0x02 | **Number of disconnected sessions**: Value will indicate the total number of disconnections from the server since startup. |
| TERMSRV\_RECON\_SESSIONS  0x03 | **Number of reconnected sessions**: Value will indicate the total number of all reconnected sessions that have existed on the server since startup. |
| TERMSRV\_CURRENT\_ACTIVE\_SESSIONS  0x04 | **Current number of active sessions**: Value will indicate the current number of active sessions on the server. |
| TERMSRV\_CURRENT\_DISC\_SESSIONS  0x05 | **Current number of disconnected sessions**: Value will indicate the current number of disconnected sessions on the server. |
| TERMSRV\_PENDING\_SESSIONS  0x06 | **Current number of pending sessions**: Value will indicate the current number of pending connections to the server.[<56>](#Appendix_A_56" \o "Product behavior note 56) |
| TERMSRV\_SUCC\_TOTAL\_LOGONS  0x07 | **Total number of successful logons**: Value will indicate the total number of successful logons on the server, both locally and remotely.[<57>](#Appendix_A_57" \o "Product behavior note 57) |
| TERMSRV\_SUCC\_LOCAL\_LOGONS  0x08 | **Total number of successful local logons**: Value will indicate the total number of successful local logons on the server.[<58>](#Appendix_A_58" \o "Product behavior note 58) |
| TERMSRV\_SUCC\_REMOTE\_LOGONS  0x09 | **Total number of successful remote logons**: Value will indicate the total number of successful remote logons on the server.[<59>](#Appendix_A_59" \o "Product behavior note 59) |
| TERMSRV\_SUCC\_SESSION0\_LOGONS  0x0A | **Total number of successful session 0 logons**: Value will indicate the total number of successful connects on the server to session 0.[<60>](#Appendix_A_60" \o "Product behavior note 60) |
| TERMSRV\_CURRENT\_TERMINATING\_SESSIONS  0x0B | **Number of terminating sessions**: Value will indicate the current number of terminating sessions on the server.[<61>](#Appendix_A_61" \o "Product behavior note 61) |
| TERMSRV\_CURRENT\_LOGGEDON\_SESSIONS  0x0C | **Number of logged on sessions**: Value will indicate the current number of logged-on sessions on the server.[<62>](#Appendix_A_62" \o "Product behavior note 62) |

**bResult:**  Set to TRUE if counter information is returned. Set to FALSE if counter data isn't being returned because the counter ID being requested was unrecognized.

#### USERCONFIG

For a specific [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) session, the **USERCONFIG** structure indicates the user and session configuration.

1. typedef struct \_USERCONFIG {
2. ULONG fInheritAutoLogon : 1;
3. ULONG fInheritResetBroken : 1;
4. ULONG fInheritReconnectSame : 1;
5. ULONG fInheritInitialProgram : 1;
6. ULONG fInheritCallback : 1;
7. ULONG fInheritCallbackNumber : 1;
8. ULONG fInheritShadow : 1;
9. ULONG fInheritMaxSessionTime : 1;
10. ULONG fInheritMaxDisconnectionTime : 1;
11. ULONG fInheritMaxIdleTime : 1;
12. ULONG fInheritAutoClient : 1;
13. ULONG fInheritSecurity : 1;
14. ULONG fPromptForPassword : 1;
15. ULONG fResetBroken : 1;
16. ULONG fReconnectSame : 1;
17. ULONG fLogonDisabled : 1;
18. ULONG fWallPaperDisabled : 1;
19. ULONG fAutoClientDrives : 1;
20. ULONG fAutoClientLpts : 1;
21. ULONG fForceClientLptDef : 1;
22. ULONG fRequireEncryption : 1;
23. ULONG fDisableEncryption : 1;
24. ULONG fUnused1 : 1;
25. ULONG fHomeDirectoryMapRoot : 1;
26. ULONG fUseDefaultGina : 1;
27. ULONG fCursorBlinkDisabled : 1;
28. ULONG fPublishedApp : 1;
29. ULONG fHideTitleBar : 1;
30. ULONG fMaximize : 1;
31. ULONG fDisableCpm : 1;
32. ULONG fDisableCdm : 1;
33. ULONG fDisableCcm : 1;
34. ULONG fDisableLPT : 1;
35. ULONG fDisableClip : 1;
36. ULONG fDisableExe : 1;
37. ULONG fDisableCam : 1;
38. ULONG fDisableAutoReconnect : 1;
39. ULONG ColorDepth : 3;
40. ULONG fInheritColorDepth: 1;
41. ULONG fErrorInvalidProfile : 1;
42. ULONG fPasswordIsScPin: 1;
43. ULONG fDisablePNPRedir:1;
44. WCHAR UserName[ USERNAME\_LENGTH + 1 ];
45. WCHAR Domain[ DOMAIN\_LENGTH + 1 ];
46. WCHAR Password[ PASSWORD\_LENGTH + 1 ];
47. WCHAR WorkDirectory[ DIRECTORY\_LENGTH + 1 ];
48. WCHAR InitialProgram[ INITIALPROGRAM\_LENGTH + 1 ];
49. WCHAR CallbackNumber[ CALLBACK\_LENGTH + 1 ];
50. CALLBACKCLASS Callback;
51. SHADOWCLASS Shadow;
52. ULONG MaxConnectionTime;
53. ULONG MaxDisconnectionTime;
54. ULONG MaxIdleTime;
55. ULONG KeyboardLayout;
56. BYTE MinEncryptionLevel;
57. WCHAR NWLogonServer[ NASIFILESERVER\_LENGTH + 1 ];
58. APPLICATIONNAME PublishedName;
59. WCHAR WFProfilePath[ DIRECTORY\_LENGTH + 1 ];
60. WCHAR WFHomeDir[ DIRECTORY\_LENGTH + 1 ];
61. WCHAR WFHomeDirDrive[ 4 ];
62. } USERCONFIG, \* PUSERCONFIG;

**fInheritAutoLogon:**  The prompt for the password setting. TRUE indicates the use of client-specified autologon settings, FALSE specifies the use of machine autologon settings.

**fInheritResetBroken:**  Reset the session when the connection is broken. TRUE indicates the value to use for **fResetBroken** from the user properties if the machine/user policy is not set, FALSE otherwise.

**fInheritReconnectSame:**  Reconnect from the same client setting. TRUE indicates the value to use for **fReconnectSame** from the user properties if the machine/user policy is not set, FALSE otherwise.

**fInheritInitialProgram:**  The initial program setting. TRUE indicates the value to use for **InitialProgram** from the user properties if the machine/user policy is not set, FALSE otherwise.

**fInheritCallback:**   The callback setting. TRUE indicates the value to use for **Callback** from the user properties if the machine/user policy is not set, FALSE otherwise.[<63>](#Appendix_A_63" \o "Product behavior note 63)

**fInheritCallbackNumber:**  The callback number setting. TRUE indicates the value to use for **CallbackNumber** from the user properties if the machine/user policy is not set, FALSE otherwise.[<64>](#Appendix_A_64" \o "Product behavior note 64)

**fInheritShadow:**  The shadow setting. TRUE indicates the value to use for **Shadow** from the user properties if the machine/user policy is not set, FALSE otherwise.

**fInheritMaxSessionTime:**  The maximum allowed session connection time setting. TRUE indicates the value to use for **MaxSessionTime** from the user properties if the machine/user policy is not set, FALSE otherwise.

**fInheritMaxDisconnectionTime:**  The maximum allowed session disconnect time setting. TRUE indicates the value to use for **MaxDisconnectionTime** from the user properties if the machine/user policy is not set, FALSE otherwise.

**fInheritMaxIdleTime:**  The maximum allowed session idle time. TRUE indicates the value to use for **MaxIdleTime** from the user properties if the machine/user policy is not set, FALSE otherwise.

**fInheritAutoClient:**  The auto client setting. TRUE indicates the value to use for **fAutoClientDrivers** and **fAutoClientLpts** from the user properties if the machine/user policy is not set, FALSE otherwise.

**fInheritSecurity:**  Inherit security setting. TRUE indicates the use of security settings from the user properties if the machine/user policy is not set, FALSE otherwise.

**fPromptForPassword:**  Set to TRUE to ignore the credential sent from the client and always prompt for a password, FALSE otherwise.

**fResetBroken:**  Set to TRUE to log off the session when the idle timers for the session expire. Otherwise, the session will be disconnected when the timer expires.

**fReconnectSame:**  FALSE indicates that the user can reconnect from any client computer to a disconnected session.

TRUE indicates that the user must reconnect to a disconnected session from the same client computer that initially established the disconnected session. Logging on from a different client computer will lead to a new terminal server session being created.

**fLogonDisabled:**  TRUE indicates that a user cannot log on to a session remotely, FALSE otherwise.[<65>](#Appendix_A_65" \o "Product behavior note 65)

**fWallPaperDisabled:**  TRUE indicates display of the desktop wallpaper in the session has been disabled, FALSE otherwise.

**fAutoClientDrives:**  TRUE specifies to automatically redirect local drives on the client so they are accessible to the user in the remote terminal server session, FALSE otherwise.

**fAutoClientLpts:**  TRUE specifies to automatically redirect printers on the client so they are accessible to the user in the remote terminal server session, FALSE otherwise.

**fForceClientLptDef:**  TRUE indicates to force the client's redirected printer to be the default printer for the user, FALSE otherwise.

**fRequireEncryption:**  TRUE indicates the connection must be encrypted, FALSE otherwise.

**fDisableEncryption:**  TRUE indicates the connection does not need encryption, FALSE otherwise.

**fUnused1:**  Not used.

**fHomeDirectoryMapRoot:**  Not used.

**fUseDefaultGina:**  TRUE indicates to override a third-party [**GINA**](#gt_23aa4520-d05c-4416-9d4b-ca86905cb426) so that only the default GINA is used for the terminal server session, FALSE otherwise.[<66>](#Appendix_A_66" \o "Product behavior note 66)

**fCursorBlinkDisabled:**  TRUE indicates disable the blinking of the mouse cursor, FALSE otherwise.[<67>](#Appendix_A_67" \o "Product behavior note 67)

**fPublishedApp:**  Not used.

**fHideTitleBar:**  Not used.

**fMaximize:**  Not used.

**fDisableCpm:**  TRUE indicates disable client printer redirection, FALSE otherwise.

**fDisableCdm:**  TRUE indicates disable client drive redirection, FALSE otherwise.

**fDisableCcm:**  TRUE indicates disable client COM port redirection, FALSE otherwise.

**fDisableLPT:**  TRUE indicates disable client printer (LPT) port redirection, FALSE otherwise.

**fDisableClip:**  TRUE indicates disable client clipboard redirection, FALSE otherwise.

**fDisableExe:**  TRUE indicates disable .exe file execution, FALSE otherwise.

**fDisableCam:**  TRUE indicates disable client audio redirection, FALSE otherwise.

**fDisableAutoReconnect:**  TRUE indicates disable auto-reconnect functionality, FALSE otherwise.[<68>](#Appendix_A_68" \o "Product behavior note 68)

**ColorDepth:**  The color depth of the session.[<69>](#Appendix_A_69" \o "Product behavior note 69)

**fInheritColorDepth:**  Set to TRUE to inherit color depth from the user or client configuration, FALSE otherwise.[<70>](#Appendix_A_70" \o "Product behavior note 70)

**fErrorInvalidProfile:**  Set to TRUE if WFProfilePath, WFHomeDir, or WFHomeDirDrive is invalid (too long), FALSE otherwise.[<71>](#Appendix_A_71" \o "Product behavior note 71)

**fPasswordIsScPin:**  Set to TRUE if the password field contains a smart card PIN.[<72>](#Appendix_A_72" \o "Product behavior note 72)

**fDisablePNPRedir:**  Set to TRUE if Plug and Play (PnP) redirection is disabled, FALSE otherwise.

**UserName:**  The user name used in autologon scenarios.

**Domain:**  The [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) name used in autologon scenarios.

**Password:**  The password used in autologon scenarios.

**WorkDirectory:**  The work directory for the initial program.

**InitialProgram:**  The program to run instead of the default.[<73>](#Appendix_A_73" \o "Product behavior note 73)

**CallbackNumber:**  The telephone number that will be returned by the [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) server to the client when the server is unable to complete the connection request from the client. The user on the client side can use this number to call back for technical support.[<74>](#Appendix_A_74" \o "Product behavior note 74)

**Callback:**  The callback class for callback operations.[<75>](#Appendix_A_75" \o "Product behavior note 75)

**Shadow:**  The shadow setting of the session.

**MaxConnectionTime:**  The maximum allowed session connection time setting of the session in milliseconds. The session will disconnect/logoff once the limit is reached.

**MaxDisconnectionTime:**  The maximum allowed session disconnect time of the session in milliseconds. The session will logoff once the limit is reached.

**MaxIdleTime:**  The maximum allowed session idle time setting of the session in milliseconds. The session will disconnect/logoff once the limit is reached.

**KeyboardLayout:**  The keyboard layout (HKL) of the session.

**MinEncryptionLevel:**  The minimum allowed encryption level. Possible numeric values for this parameter include 1 (Low), 2 (Client Compatible), 3 (High), and 4 (FIPS). Detailed description of these encryption levels is included in [[MS-RDPBCGR]](%5bMS-RDPBCGR%5d.pdf#Section_5073f4ed1e9345e1b0396e30c385867c) sections 5.3.1 and 5.4.1.

**NWLogonServer:**  The NetWare logon server name.[<76>](#Appendix_A_76" \o "Product behavior note 76)

**PublishedName:**  Not used.

**WFProfilePath:**  The terminal server profile path. Overrides the standard profile path.

**WFHomeDir:**  The terminal server home directory path. Overrides the standard home directory.

**WFHomeDirDrive:**  The terminal server home directory drive. Overrides the standard home directory.

##### CALLBACKCLASS

The **CALLBACKCLASS** enumeration is used for callback options to indicate the type of callback.[<77>](#Appendix_A_77" \o "Product behavior note 77)

1. typedef enum \_CALLBACKCLASS
2. {
3. Callback\_Disable,
4. Callback\_Roving,
5. Callback\_Fixed,
6. } CALLBACKCLASS;

**Callback\_Disable:** Callback is disabled.

**Callback\_Roving:** The callback number is a roving number.

**Callback\_Fixed:** The callback number is a fixed number.

##### APPLICATIONNAME

This data type represents an application name.

1. typedef WCHAR APPLICATIONNAME[MAX\_BR\_NAME];
2. typedef WCHAR\* PAPPLICATIONNAME;

#### WINSTATIONCLIENT

The WINSTATIONCLIENT structure defines the client-requested configuration when connecting to a session.

1. typedef struct \_WINSTATIONCLIENT {
2. ULONG fTextOnly  :1;
3. ULONG fDisableCtrlAltDel  :1;
4. ULONG fMouse  :1;
5. ULONG fDoubleClickDetect  :1;
6. ULONG fINetClient  :1;
7. ULONG fPromptForPassword  :1;
8. ULONG fMaximizeShell  :1;
9. ULONG fEnableWindowsKey  :1;
10. ULONG fRemoteConsoleAudio  :1;
11. ULONG fPasswordIsScPin  :1;
12. ULONG fNoAudioPlayback  :1;
13. ULONG fUsingSavedCreds  :1;
14. ULONG fRestrictedLogon  :1;
15. WCHAR ClientName[CLIENTNAME\_LENGTH + 1 ];
16. WCHAR Domain[DOMAIN\_LENGTH + 1 ];
17. WCHAR UserName[USERNAME\_LENGTH + 1 ];
18. WCHAR Password[PASSWORD\_LENGTH + 1 ];
19. WCHAR WorkDirectory[DIRECTORY\_LENGTH + 1 ];
20. WCHAR InitialProgram[INITIALPROGRAM\_LENGTH + 1 ];
21. ULONG SerialNumber;
22. BYTE EncryptionLevel;
23. ULONG ClientAddressFamily;
24. WCHAR ClientAddress[CLIENTADDRESS\_LENGTH + 1 ];
25. USHORT HRes;
26. USHORT VRes;
27. USHORT ColorDepth;
28. USHORT ProtocolType;
29. ULONG KeyboardLayout;
30. ULONG KeyboardType;
31. ULONG KeyboardSubType;
32. ULONG KeyboardFunctionKey;
33. WCHAR imeFileName[IMEFILENAME\_LENGTH + 1 ];
34. WCHAR ClientDirectory[DIRECTORY\_LENGTH + 1 ];
35. WCHAR ClientLicense[CLIENTLICENSE\_LENGTH + 1 ];
36. WCHAR ClientModem[CLIENTMODEM\_LENGTH + 1 ];
37. ULONG ClientBuildNumber;
38. ULONG ClientHardwareId;
39. USHORT ClientProductId;
40. USHORT OutBufCountHost;
41. USHORT OutBufCountClient;
42. USHORT OutBufLength;
43. WCHAR AudioDriverName[9];
44. TS\_TIME\_ZONE\_INFORMATION ClientTimeZone;
45. ULONG ClientSessionId;
46. WCHAR clientDigProductId[CLIENT\_PRODUCT\_ID\_LENGTH];
47. ULONG PerformanceFlags;
48. ULONG ActiveInputLocale;
49. } WINSTATIONCLIENT,
50. \*PWINSTATIONCLIENT;

**fTextOnly:**   Text-only client session. This is always FALSE.

**fDisableCtrlAltDel:**   Set to TRUE to specify that CTRL+ALT+DEL is disabled.

**fMouse:**   TRUE indicates the mouse is connected to the client, FALSE otherwise.

**fDoubleClickDetect:**   Double-click the detect flag. TRUE indicates detect double-click, FALSE otherwise.

**fINetClient:**   Always set to FALSE.

**fPromptForPassword:**   TRUE indicates the user will always be prompted for a password, even if the password is saved from previous connection; FALSE otherwise.

**fMaximizeShell:**   TRUE indicates maximize the shell, FALSE otherwise.

**fEnableWindowsKey:**   TRUE indicates that the Windows key (E0\_5B) is enabled in the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) session. FALSE indicates that it is disabled.

**fRemoteConsoleAudio:**  Set to TRUE if audio for the console session is left remotely at the server, FALSE otherwise.[<78>](#Appendix_A_78" \o "Product behavior note 78)

**fPasswordIsScPin:**  Set to TRUE if the password field contains a smart card PIN, FALSE otherwise.[<79>](#Appendix_A_79" \o "Product behavior note 79)

**fNoAudioPlayback:**  Set to TRUE to disable audio playback, or FALSE to enable audio playback.[<80>](#Appendix_A_80" \o "Product behavior note 80)

**fUsingSavedCreds:**   Set to TRUE if the terminal server connection was made using a credential saved on the client computer, FALSE otherwise.

**fRestrictedLogon:**  Set to TRUE if the client is running in Restricted Administration mode, FALSE otherwise. In Restricted Administration mode, user credentials are not sent to the server, which can protect the user if the server has been compromised.[<81>](#Appendix_A_81" \o "Product behavior note 81)

**ClientName:**   The name of the client computer.

**Domain:**   The user's [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) name.

**UserName:**   The user's user name.

**Password:**   The user's password.

**WorkDirectory:**   The work directory for the initial program.

**InitialProgram:**  The program to run instead of the default.[<82>](#Appendix_A_82" \o "Product behavior note 82)

**SerialNumber:**   The client computer's unique serial number.

**EncryptionLevel:**   The encryption level.

**ClientAddressFamily:**  The address family of the client's address.[<83>](#Appendix_A_83" \o "Product behavior note 83)

**ClientAddress:**   The client's address. The format depends on the value of **ClientAddressFamily**. See [[MSDN-SOCKET]](https://go.microsoft.com/fwlink/?LinkId=93442) for more information.

**HRes:**   The horizontal resolution, in pixels.

**VRes:**   The vertical resolution, in pixels.

**ColorDepth:**  The color depth. [<84>](#Appendix_A_84" \o "Product behavior note 84)

**ProtocolType:**  The type of protocol.[<85>](#Appendix_A_85" \o "Product behavior note 85)

**KeyboardLayout:**   The keyboard layout (HKL).

**KeyboardType:**   The keyboard type.

**KeyboardSubType:**   The keyboard subtype.

**KeyboardFunctionKey:**   The number of keyboard function keys.

**imeFileName:**   The file name of the [**input method editor (IME)**](#gt_6700555e-25ae-4c2f-b775-419d5f532d8e), if any, used for the session. For more information on IMEs, see [[MSFT-IME]](https://go.microsoft.com/fwlink/?LinkId=100302).

**ClientDirectory:**   The directory in which the client was installed.

**ClientLicense:**  The client's license.[<86>](#Appendix_A_86" \o "Product behavior note 86)

**ClientModem:**  The client's modem.[<87>](#Appendix_A_87" \o "Product behavior note 87)

**ClientBuildNumber:**   The client's build number.

**ClientHardwareId:**   The client-specific hardware identifier.

**ClientProductId:**   The client-specific product identifier.

**OutBufCountHost:**   The number of output buffers on the host computer.

**OutBufCountClient:**   The number of output buffers on the client computer.

**OutBufLength:**   The length of the output buffer, in bytes.

**AudioDriverName:**   The audio driver's name.

**ClientTimeZone:**  The client's time zone.[<88>](#Appendix_A_88" \o "Product behavior note 88)

**ClientSessionId:**  The client's session ID.[<89>](#Appendix_A_89" \o "Product behavior note 89)

**clientDigProductId:**  The client-specific product ID.[<90>](#Appendix_A_90" \o "Product behavior note 90)

**PerformanceFlags:**  Protocol-specific performance flags.[<91>](#Appendix_A_91" \o "Product behavior note 91) It MUST be any bitwise OR combination of the following except **TS\_PERF\_DISABLE\_NOTHING**.

| Value | Meaning |
| --- | --- |
| TS\_PERF\_DISABLE\_NOTHING  0x00000000 | Disable nothing. |
| TS\_PERF\_DISABLE\_WALLPAPER  0x00000001 | Disable wallpaper. |
| TS\_PERF\_DISABLE\_FULLWINDOWDRAG  0x00000002 | Disable full window drag animation. |
| TS\_PERF\_DISABLE\_MENUANIMATIONS  0x00000004 | Disable menu animations. |
| TS\_PERF\_DISABLE\_THEMING  0x00000008 | Disable themes. |
| TS\_PERF\_ENABLE\_ENHANCED\_GRAPHICS  0x00000010 | Enable enhanced graphics. |
| TS\_PERF\_DISABLE\_CURSOR\_SHADOW  0x00000020 | Disable cursor shadow. |
| TS\_PERF\_DISABLE\_CURSORSETTINGS  0x00000040 | Disable cursor settings. |
| TS\_PERF\_ENABLE\_FONT\_SMOOTHING  0x00000080 | Enable font smoothing.[<92>](#Appendix_A_92" \o "Product behavior note 92) |
| TS\_PERF\_ENABLE\_DESKTOP\_COMPOSITION  0x00000100 | Enable desktop composition.[<93>](#Appendix_A_93" \o "Product behavior note 93) |
| TS\_PERF\_DEFAULT\_NONPERFCLIENT\_SETTING  0x40000000 | Reserved and used internally by the client. |
| TS\_PERF\_RESERVED1  0x80000000 | Reserved and used internally by the client. |

**ActiveInputLocale:**  Client language locale HKL.[<94>](#Appendix_A_94" \o "Product behavior note 94)

For information about keyboard functions and handling, see [MSFT-W2KDDK].

##### TS\_TIME\_ZONE\_INFORMATION

The TS\_TIME\_ZONE\_INFORMATION structure contains client time zone information.

1. typedef struct \_TS\_TIME\_ZONE\_INFORMATION {
2. LONG Bias;
3. WCHAR StandardName[32 ];
4. TS\_SYSTEMTIME StandardDate;
5. LONG StandardBias;
6. WCHAR DaylightName[32 ];
7. TS\_SYSTEMTIME DaylightDate;
8. LONG DaylightBias;
9. } TS\_TIME\_ZONE\_INFORMATION;

**Bias:**  A 32-bit integer. Current bias for local time translation on the client, in minutes. The bias is the difference, in minutes, between Coordinated Universal Time (UTC) and local time. All translations between UTC and local time are based on the following formula:

UTC = local time + bias

**StandardName:**  A description for standard time on the client. For example, this field could contain the string "Pacific Standard Time" to indicate Pacific Standard Time. An array of 32 [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) characters.

**StandardDate:**  A TS\_SYSTEMTIME structure that contains the date and local time when the transition from daylight saving time to standard time occurs on the client. If this field is specified, the **DaylightDate** field is also specified.

**StandardBias:**  A 32-bit integer that defines the bias value in number of minutes to be used during local time translations that occur during standard time. This field SHOULD be ignored if a value is not supplied in the **StandardDate** field. This value is added to the value of the **Bias** field to form the bias used during standard time. In most time zones, the value of this field is 0.

**DaylightName:**  An array of 32 Unicode characters that describes daylight time on the client. For example, this field could contain "Pacific Daylight Time" to indicate Pacific Daylight Time.

**DaylightDate:**  A TS\_SYSTEMTIME that contains a date and local time when the transition from standard time to daylight saving time occurs on the client. If this field is specified, the **StandardDate** field is also specified.

**DaylightBias:**  A 32-bit integer that defines the bias value to be used during local time translations that occur during daylight saving time. This field SHOULD be ignored if a value for the **DaylightDate** field is not supplied. This value is added to the value of the **Bias** field to form the bias used during daylight saving time. In most time zones, the value of this field is 60.

###### TS\_SYSTEMTIME

Information about a time zone. This structure is identical to the structure SYSTEMTIME. For more information, see [[MSDN-SYSTIME]](https://go.microsoft.com/fwlink/?LinkId=93364).

1. typedef struct \_TS\_SYSTEMTIME {
2. USHORT wYear;
3. USHORT wMonth;
4. USHORT wDayOfWeek;
5. USHORT wDay;
6. USHORT wHour;
7. USHORT wMinute;
8. USHORT wSecond;
9. USHORT wMilliseconds;
10. } TS\_SYSTEMTIME;

**wYear:**  The year when transition from daylight saving time to standard time occurs (1601 to 30827).

**wMonth:**  The month when transition from daylight saving time to standard time occurs.

This member can be one of the following values.

| Value | Meaning |
| --- | --- |
| 1 | January |
| 2 | February |
| 3 | March |
| 4 | April |
| 5 | May |
| 6 | June |
| 7 | July |
| 8 | August |
| 9 | September |
| 10 | October |
| 11 | November |
| 12 | December |

**wDayOfWeek:**   The day of the week when the transition from daylight saving time to standard time occurs.

This member can be one of the following values.

| Value | Meaning |
| --- | --- |
| 0 | Sunday |
| 1 | Monday |
| 2 | Tuesday |
| 3 | Wednesday |
| 4 | Thursday |
| 5 | Friday |
| 6 | Saturday |

**wDay:**   The occurrence of **wDayOfWeek** within the month when the transition from daylight saving time to standard time takes place.

This member can be one of the following values.

| Value | Meaning |
| --- | --- |
| 1 | First occurrence of **wDayOfWeek** |
| 2 | Second occurrence of **wDayOfWeek** |
| 3 | Third occurrence of **wDayOfWeek** |
| 4 | Fourth occurrence of **wDayOfWeek** |
| 5 | Last occurrence of **wDayOfWeek** |

**wHour:**   The hour when transition from daylight saving time to standard time occurs (0 to 23).

**wMinute:**   The minute when transition from daylight saving time to standard time occurs (0 to 59).

**wSecond:**   The second when transition from daylight saving time to standard time occurs (0 to 59).

**wMilliseconds:**   The millisecond when transition from daylight saving time to standard time occurs (0 to 999).

#### WINSTATIONINFORMATION

Provides the current values of various properties such as state, connect time, last input time, and so on, for a session.

1. typedef struct \_WINSTATIONINFORMATION {
2. WINSTATIONSTATECLASS ConnectState;
3. WINSTATIONNAME WinStationName;
4. ULONG LogonId;
5. LARGE\_INTEGER ConnectTime;
6. LARGE\_INTEGER DisconnectTime;
7. LARGE\_INTEGER LastInputTime;
8. LARGE\_INTEGER LogonTime;
9. PROTOCOLSTATUS Status;
10. WCHAR Domain[DOMAIN\_LENGTH + 1];
11. WCHAR UserName[USERNAME\_LENGTH + 1];
12. LARGE\_INTEGER CurrentTime;
13. } WINSTATIONINFORMATION,
14. \*PWINSTATIONINFORMATION;

**ConnectState:**  The current connect state of the session.

**WinStationName:**  The name of the session.

**LogonId:**  The session identifier of the session.

**ConnectTime:**  The time of the most recent connection to the session. This is a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601 (UTC).

**DisconnectTime:**  The time of the most recent disconnection from the session. This is a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601 (UTC).

**LastInputTime:**  The time the session last received input. This is an indicator of how long a session has been idle. This is a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601 (UTC).

**LogonTime:**  The time of the logon to the session. This is a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601 (UTC).

**Status:**  The status of the protocol, as specified in section [2.2.2.20.1](#Section_c9066753acbd46789a728fb1b080bd09).

**Domain:**  The user's [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) name.

**UserName:**  The user's user name.

**CurrentTime:**  The current time in the session. This is a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601 (UTC).

##### PROTOCOLSTATUS

The status of the protocol used by the session.

1. typedef struct \_PROTOCOLSTATUS {
2. PROTOCOLCOUNTERS Output;
3. PROTOCOLCOUNTERS Input;
4. CACHE\_STATISTICS Cache;
5. ULONG AsyncSignal;
6. ULONG AsyncSignalMask;
7. } PROTOCOLSTATUS,
8. \*PPROTOCOLSTATUS;

**Output:**   A [PROTOCOLCOUNTERS](#Section_c5d7f335bfd9444d87a5b0d028a52b5e) structure containing the output protocol counters.

**Input:**   A PROTOCOLCOUNTERS structure containing the input protocol counters.

**Cache:**   A [CACHE\_STATISTICS](#Section_81203ca2e58b4681affa924e59671b5c) structure containing statistics for the cache.

**AsyncSignal:**   Indicator of async signal, such as MS\_CTS\_ON, for async protocols. For more information on asynchronous protocols, see [[MSDN-SERIAL]](https://go.microsoft.com/fwlink/?LinkId=100301).

**AsyncSignalMask:**   Mask of async signal events, such as EV\_CTS, for async protocols. For more information on asynchronous protocols, see [MSDN-SERIAL].

###### PROTOCOLSTATUSEX

The **PROTOCOLSTATUSEX** structure defines the extended status of the protocol used by the session.

1. typedef struct {
2. PROTOCOLSTATUS ProtocolStatus;
3. LARGE\_INTEGER Counters[MAX\_COUNTER\_EXTENSIONS];
4. } PROTOCOLSTATUSEX,
5. \*PPROTOCOLSTATUSEX;

**ProtocolStatus:**  The status of the protocol as described in section [2.2.2.20.1](#Section_c9066753acbd46789a728fb1b080bd09).

**Counters:**  The value of the various counters associated with the protocol as specified in [PROTOCOLCOUNTERS](#Section_c5d7f335bfd9444d87a5b0d028a52b5e).

###### PROTOCOLCOUNTERS

Protocol performance counters.

1. typedef struct \_PROTOCOLCOUNTERS {
2. ULONG WdBytes;
3. ULONG WdFrames;
4. ULONG WaitForOutBuf;
5. ULONG Frames;
6. ULONG Bytes;
7. ULONG CompressedBytes;
8. ULONG CompressFlushes;
9. ULONG Errors;
10. ULONG Timeouts;
11. ULONG AsyncFramingError;
12. ULONG AsyncOverrunError;
13. ULONG AsyncOverflowError;
14. ULONG AsyncParityError;
15. ULONG TdErrors;
16. USHORT ProtocolType;
17. USHORT Length;
18. union {
19. TSHARE\_COUNTERS TShareCounters;
20. ULONG Reserved[100];
21. } Specific;
22. } PROTOCOLCOUNTERS,
23. \*PPROTOCOLCOUNTERS;

**WdBytes:**  [**WinStation**](#gt_8cc89bb9-be94-4672-8a5d-9947b91bc952) driver number of bytes sent and received.

**WdFrames:**  WinStation driver number of frames sent and received.

**WaitForOutBuf:**  The number of times waited for an output buffer to become available.

**Frames:**  Transport driver number of frames.

**Bytes:**  Transport driver number of bytes.

**CompressedBytes:**  Number of compressed bytes.

**CompressFlushes:**  Number of compress flushes. A compress flush occurs when compression for a packet fails and the original uncompressed packet replaces it.

**Errors:**  Number of packets that were in error during the session.

**Timeouts:**  Number of time-outs.

**AsyncFramingError:**  Number of async framing errors.

**AsyncOverrunError:**  Number of async overrun errors.

**AsyncOverflowError:**  Number of async overflow errors.

**AsyncParityError:**  Number of async parity errors.

**TdErrors:**  Number of transport protocol errors.

**ProtocolType:**  Protocol type.

**Length:**  Length of data in the protocol-specific area. Can be up to 100 \* sizeof(ULONG) in size.

**Specific:**  Specifies which types of counters are to be queried. It can be one of the following:

**TShareCounters:**  Protocol performance counters.

**Reserved:**  Reserved for future use.

TSHARE\_COUNTERS

TSHARE\_COUNTERS is not used.

1. typedef struct \_TSHARE\_COUNTERS {
2. ULONG Reserved;
3. } TSHARE\_COUNTERS,
4. \*PTSHARE\_COUNTERS;

**Reserved:**  This value is not used.

###### CACHE\_STATISTICS

Cache statistics on the protocol.

1. typedef struct CACHE\_STATISTICS {
2. USHORT ProtocolType;
3. USHORT Length;
4. union {
5. RESERVED\_CACHE ReservedCacheStats;
6. TSHARE\_CACHE TShareCacheStats;
7. ULONG Reserved[20];
8. } Specific;
9. } CACHE\_STATISTICS,
10. \*PCACHE\_STATISTICS;

**ProtocolType:**  Protocol type.

**Length:**  Length of data in the protocol-specific area. Can be up to 20 \* sizeof(ULONG) in size.

**Specific:**  The union of the following members:

**ReservedCacheStats:**  Not used.

**TShareCacheStats:**  Protocol cache statistics.

**Reserved:**  Reserved for future use.

RESERVED\_CACHE

Cache statistics.[<95>](#Appendix_A_95" \o "Product behavior note 95)

1. typedef struct \_RESERVED\_CACHE {
2. THINWIRECACHE ThinWireCache[MAX\_THINWIRECACHE ];
3. } RESERVED\_CACHE,
4. \*PRESERVED\_CACHE;

**ThinWireCache:**  The ThinWireCache structure used for the terminal server's display for RESERVED\_CACHE.

THINWIRECACHE

The ThinWireCache structure used for the terminal server's display for RESERVED\_CACHE.

1. typedef struct \_THINWIRECACHE {
2. ULONG CacheReads;
3. ULONG CacheHits;
4. } THINWIRECACHE,
5. \*PTHINWIRECACHE;

**CacheReads:**  Number of cache reads.

**CacheHits:**  Number of cache hits.

TSHARE\_CACHE

TSHARE\_CACHE is not used.

1. typedef struct \_TSHARE\_CACHE {
2. ULONG Reserved;
3. } TSHARE\_CACHE,
4. \*PTSHARE\_CACHE;

**Reserved:**  This value is not used.

#### PDPARAMS

The protocol driver parameters structure. The core [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) binaries only read this data from the system data store and pass it to callers of [RpcWinStationQueryInformation](#Section_1bba9ff271d349a3bb262e5f6fcab3ee) and other places. The core Terminal Services binaries do not process this data in any way before returning it to callers. The actual use of this configuration data is in lower-level protocol drivers.

1. typedef struct \_PDPARAMS {
2. SDCLASS SdClass;
3. union {
4. NETWORKCONFIG Network;
5. ASYNCCONFIG Async;
6. NASICONFIG Nasi;
7. OEMTDCONFIG OemTd;
8. };
9. } PDPARAMS,
10. \*PPDPARAMS;

**SdClass:**  Stack driver class. Indicates which one of the union's structures is valid.

**Network:**  Configuration of network drivers. Used if **SdClass** is SdNetwork.

**Async:**  Configuration of async (modem) driver. Used if **SdClass** is SdAsync.[<96>](#Appendix_A_96" \o "Product behavior note 96)

**Nasi:**  Reserved.

**OemTd:**  Configuration of OEM transport driver. Used if **SdClass** is SdOemTransport.[<97>](#Appendix_A_97" \o "Product behavior note 97)

#### NETWORKCONFIG

The network protocol driver's configuration structure. The following block determines NETWORKCONFIG.

1. typedef struct \_NETWORKCONFIG {
2. LONG LanAdapter;
3. DEVICENAME NetworkName;
4. ULONG Flags;
5. } NETWORKCONFIG,
6. \*PNETWORKCONFIG;

**LanAdapter:**  The LANA ID of the network adapter.

**NetworkName:**  Not used.

**Flags:**  Not used.

#### ASYNCCONFIG

The asynchronous protocol driver's configuration structure.

1. typedef struct \_ASYNCCONFIG {
2. DEVICENAME DeviceName;
3. MODEMNAME ModemName;
4. ULONG BaudRate;
5. ULONG Parity;
6. ULONG StopBits;
7. ULONG ByteSize;
8. ULONG fEnableDsrSensitivity  :1;
9. ULONG fConnectionDriver  :1;
10. FLOWCONTROLCONFIG FlowControl;
11. CONNECTCONFIG Connect;
12. } ASYNCCONFIG,
13. \*PASYNCCONFIG;

**DeviceName:**  The device's name.

**ModemName:**  The modem's name.

**BaudRate:**  The baud rate of the modem.

**Parity:**  The parity setting.

**StopBits:**  The number of stop bits.

**ByteSize:**  The size of a byte.

**fEnableDsrSensitivity:**  TRUE indicates enable Data Set Ready (DSR) sensitivity, FALSE otherwise.

**fConnectionDriver:**  Set to TRUE if there is a connection driver, FALSE otherwise.

**FlowControl:**  The flow control setting of the modem.

**Connect:**  The connect configuration.

##### MODEMNAME

The name of a modem.

1. typedef WCHAR MODEMNAME[MODEMNAME\_LENGTH + 1];
2. typedef WCHAR\* PMODEMNAME;

##### FLOWCONTROLCONFIG

Flow control configuration of an async driver.

1. typedef struct \_FLOWCONTROLCONFIG {
2. ULONG fEnableSoftwareTx  :1;
3. ULONG fEnableSoftwareRx  :1;
4. ULONG fEnableDTR  :1;
5. ULONG fEnableRTS  :1;
6. CHAR XonChar;
7. CHAR XoffChar;
8. FLOWCONTROLCLASS Type;
9. RECEIVEFLOWCONTROLCLASS HardwareReceive;
10. TRANSMITFLOWCONTROLCLASS HardwareTransmit;
11. } FLOWCONTROLCONFIG,
12. \*PFLOWCONTROLCONFIG;

**fEnableSoftwareTx:**  TRUE indicates software transmit flow control, FALSE otherwise.

**fEnableSoftwareRx:**  TRUE indicates software receive flow control, FALSE otherwise.

**fEnableDTR:**  TRUE indicates Data Terminal Ready (DTR) enabled, FALSE otherwise.

**fEnableRTS:**  TRUE indicates Request to Send (RTS) enabled, FALSE otherwise.

**XonChar:**  Xon flow control character.

**XoffChar:**  Xoff flow control character.

**Type:**  The type of flow control in use.

**HardwareReceive:**  Hardware receive flow control information.

**HardwareTransmit:**  Hardware transmit flow control information.

###### FLOWCONTROLCLASS

The FLOWCONTROLCLASS enumeration specifies the type of flow control, if any, supported.

1. typedef enum \_FLOWCONTROLCLASS
2. {
3. FlowControl\_None,
4. FlowControl\_Hardware,
5. FlowControl\_Software
6. } FLOWCONTROLCLASS;

**FlowControl\_None:** Flow control is not enabled.

**FlowControl\_Hardware:** Hardware flow control is enabled.

**FlowControl\_Software:** Software flow control is enabled.

###### RECEIVEFLOWCONTROLCLASS

The **RECEIVEFLOWCONTROLCLASS** enumeration specifies which, if any, means of receive flow control are supported.

1. typedef enum \_RECEIVEFLOWCONTROLCLASS
2. {
3. ReceiveFlowControl\_None,
4. ReceiveFlowControl\_RTS,
5. ReceiveFlowControl\_DTR,
6. } RECEIVEFLOWCONTROLCLASS;

**ReceiveFlowControl\_None:** No receive flow control currently.

**ReceiveFlowControl\_RTS:** Receive flow control Request to Send (RTS).

**ReceiveFlowControl\_DTR:** Receive flow control Data Terminal Ready (DTR).

###### TRANSMITFLOWCONTROLCLASS

The **TRANSMITFLOWCONTROLCLASS** enumeration specifies which, if any, means of transit flow control is supported.

1. typedef enum \_TRANSMITFLOWCONTROLCLASS
2. {
3. TransmitFlowControl\_None,
4. TransmitFlowControl\_CTS,
5. TransmitFlowControl\_DSR,
6. } TRANSMITFLOWCONTROLCLASS;

**TransmitFlowControl\_None:** No transmit flow control currently.

**TransmitFlowControl\_CTS:** Transmit flow control Clear to Send (CTS).

**TransmitFlowControl\_DSR:** Transmit flow control Data Set Ready (DSR).

##### CONNECTCONFIG

The CONNECTCONFIG structure specifies connectivity parameters.

1. typedef struct \_CONNECTCONFIG {
2. ASYNCCONNECTCLASS Type;
3. ULONG fEnableBreakDisconnect  :1;
4. } CONNECTCONFIG,
5. \*PCONNECTCONFIG;

**Type:**  Type of asynchronous connection. This value is not used.

**fEnableBreakDisconnect:**  If TRUE, enable break disconnect.

###### ASYNCCONNECTCLASS

The **ASYNCCONNECTCLASS** enumeration is not used.

1. typedef enum \_ASYNCCONNECTCLASS {
2. Connect\_CTS,
3. Connect\_DSR,
4. Connect\_RI,
5. Connect\_DCD,
6. Connect\_FirstChar,
7. Connect\_Perm,
8. } ASYNCCONNECTCLASS;

#### NASICONFIG

The **NASICONFIG** structure determines the data type of NASICONFIG.

1. typedef struct \_NASICONFIG {
2. NASISPECIFICNAME SpecificName;
3. NASIUSERNAME UserName;
4. NASIPASSWORD PassWord;
5. NASISESIONNAME SessionName;
6. NASIFILESERVER FileServer;
7. BOOLEAN GlobalSession;
8. } NASICONFIG, \*PNASICONFIG;

**SpecificName:**  The NASI-specific (Netware Asynchronous Services Interface) name.

**UserName:**  The NASI user's user name.

**PassWord:**  The NASI user's password.

**SessionName:**  The NASI session name.

**FileServer:**  The NASI file server name.

**GlobalSession:**  Set to TRUE if the session is a global session.

##### NASIUSERNAME

The NASI user's user name.

This type is declared as follows:

1. typedef WCHAR NASIUSERNAME[NASIUSERNAME\_LENGTH + 1];

##### NASIPASSWORD

The NASI user's password.

This type is declared as follows:

1. typedef WCHAR NASIPASSWORD[NASIPASSWORD\_LENGTH + 1];

##### NASISESIONNAME

The NASI session name.

This type is declared as follows:

1. typedef WCHAR NASISESIONNAME[NASISESSIONNAME\_LENGTH + 1];

##### NASISPECIFICNAME

The NASI-specific name.

This type is declared as follows:

1. typedef WCHAR NASISPECIFICNAME[NASISPECIFICNAME\_LENGTH + 1];

##### NASIFILESERVER

The NASI file server name.

This type is declared as follows:

1. typedef WCHAR NASIFILESERVER[NASIFILESERVER\_LENGTH + 1];

#### OEMTDCONFIG

The OEM transport driver configuration structure.

1. typedef struct \_OEMTDCONFIG {
2. LONG Adapter;
3. DEVICENAME DeviceName;
4. ULONG Flags;
5. } OEMTDCONFIG,
6. \*POEMTDCONFIG;

**Adapter:**  The ID of the adapter (OEM driver-specific).

**DeviceName:**  The network name (OEM driver-specific).

**Flags:**  Driver flags (OEM driver-specific).

#### PDCONFIG

The protocol driver configuration structure.

1. typedef struct \_PDCONFIG {
2. PDCONFIG2 Create;
3. PDPARAMS Params;
4. } PDCONFIG,
5. \*PPDCONFIG;

**Create:**  The software configuration of the driver.

**Params:**  The hardware configuration for the driver.

##### PDCONFIG2

The protocol driver's software configuration.

1. typedef struct \_PDCONFIG2 {
2. PDNAME PdName;
3. SDCLASS SdClass;
4. DLLNAME PdDLL;
5. ULONG PdFlag;
6. ULONG OutBufLength;
7. ULONG OutBufCount;
8. ULONG OutBufDelay;
9. ULONG InteractiveDelay;
10. ULONG PortNumber;
11. ULONG KeepAliveTimeout;
12. } PDCONFIG2,
13. \*PPDCONFIG2;

**PdName:**  The descriptive name of the protocol driver.

**SdClass:**  The type of driver.

**PdDLL:**  The driver's image name.

**PdFlag:**  Driver flags MUST be any bitwise OR combination of the following values:

| Value | Meaning |
| --- | --- |
| PD\_UNUSED  0x00000001 | Unused. |
| PD\_RELIABLE  0x00000002 | Error-free protocol. |
| PD\_FRAME  0x00000004 | Frame-oriented protocol. |
| PD\_CONNECTION  0x00000008 | Connection-oriented protocol. |
| PD\_CONSOLE  0x00000010 | Directly connected console. |
| PD\_LANA  0x00000020 | Network class uses LANAs (NetBIOS). |
| PD\_TRANSPORT  0x00000040 | Transport driver (lowest level). |
| PD\_SINGLE\_INST  0x00000080 | Single instance only (async). |
| PD\_NOLOW\_WATERMARK  0x00000100 | Low water mark to resume transmission. |

**OutBufLength:**  Optimal output buffers length, in bytes.

**OutBufCount:**  Optimal number of output buffers.

**OutBufDelay:**  Write delay, in milliseconds.

**InteractiveDelay:**  Write delay during active input.

**PortNumber:**  Network listen port number.

**KeepAliveTimeout:**  Frequency to send keep-alives, in milliseconds.

##### PDNAME

The protocol driver name (PDNAME) data type.

1. typedef WCHAR PDNAME[PDNAME\_LENGTH + 1];
2. typedef WCHAR\* PPDNAME;

#### WDCONFIG

The [**WinStation**](#gt_8cc89bb9-be94-4672-8a5d-9947b91bc952) (session) driver configuration.[<98>](#Appendix_A_98" \o "Product behavior note 98)

1. typedef struct \_WDCONFIG {
2. WDNAME WdName;
3. DLLNAME WdDLL;
4. DLLNAME WsxDLL;
5. ULONG WdFlag;
6. ULONG WdInputBufferLength;
7. DLLNAME CfgDLL;
8. WDPREFIX WdPrefix;
9. } WDCONFIG,
10. \*PWDCONFIG;

**WdName:**  The descriptive name of the WinStation driver.

**WdDLL:**  The driver's image name.

**WsxDLL:**  Used by the [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) service to communicate with the WinStation driver.[<99>](#Appendix_A_99" \o "Product behavior note 99)

**WdFlag:**  Driver flags. It MUST be any bitwise OR combination of the following values.

| Value | Meaning |
| --- | --- |
| WDF\_UNUSED  0x00000001 | Not used. |
| WDF\_SHADOW\_SOURCE  0x00000002 | Valid shadow source. |
| WDF\_SHADOW\_TARGET  0x00000004 | Valid shadow target. |
| WDF\_OTHER  0x00000008 | Other protocol. |
| WDF\_TSHARE  0x00000010 | Remote Protocol used by Terminal Services. |
| WDF\_DYNAMIC\_RECONNECT  0x00000020 | Session can resize display at reconnect.[<100>](#Appendix_A_100" \o "Product behavior note 100) |
| WDF\_USER\_VCIOCTL  0x00000040 | User mode applications can send virtual channel IOCTL. |
| WDF\_SUBDESKTOP  0x00008000 | Sub-desktop session.[<101>](#Appendix_A_101" \o "Product behavior note 101) |

**WdInputBufferLength:**  Length, in bytes, of the input buffer used by the driver. Defaults to 2048.

**CfgDLL:**  Configuration DLL used by Terminal Services administrative tools for configuring the driver.[<102>](#Appendix_A_102" \o "Product behavior note 102)

**WdPrefix:**  Used as the prefix of the WinStation name generated for the connected sessions with this WinStation driver.[<103>](#Appendix_A_103" \o "Product behavior note 103)

##### WDNAME

The WDNAME data type.

1. typedef WCHAR WDNAME[WDNAME\_LENGTH + 1];
2. typedef WCHAR\* PWDNAME;

##### WDPREFIX

The WDPREFIX data type.

1. typedef WCHAR WDPREFIX[WDPREFIX\_LENGTH + 1];
2. typedef WCHAR\* PWDPREFIX;

#### CDCONFIG

Connection driver configuration.[<104>](#Appendix_A_104" \o "Product behavior note 104) It is used for connecting via modem to a server.

1. typedef struct \_CDCONFIG {
2. CDCLASS CdClass;
3. CDNAME CdName;
4. DLLNAME CdDLL;
5. ULONG CdFlag;
6. } CDCONFIG,
7. \*PCDCONFIG;

**CdClass:**  Connection driver type.

**CdName:**  Connection driver descriptive name.

**CdDLL:**  Connection driver image name.

**CdFlag:**  Connection driver flags. Connection driver specific.

##### CDCLASS

The **CDCLASS** enumeration specifies a type of connection driver.

1. typedef enum \_CDCLASS
2. {
3. CdNone,
4. CdModem,
5. CdClass\_Maximum,
6. } CDCLASS;

**CdNone:** No connection driver.

**CdModem:** Connection driver is a modem.

**CdClass\_Maximum:** A given CdClass variable will always be less than this value.

##### CDNAME

The **CDNAME** type contains the connection driver name.

1. typedef WCHAR CDNAME[CDNAME\_LENGTH + 1];
2. typedef WCHAR\* PCDNAME;

#### WINSTATIONCREATE

The **WINSTATIONCREATE** structure specifies a session to which the user can connect.

1. typedef struct \_WINSTATIONCREATE {
2. ULONG fEnableWinStation :1;
3. ULONG MaxInstanceCount;
4. } WINSTATIONCREATE, \* PWINSTATIONCREATE;

**fEnableWinStation:**  TRUE if enabled.

**MaxInstanceCount:**  Maximum number of instances that can connect to the [**WinStation**](#gt_8cc89bb9-be94-4672-8a5d-9947b91bc952).

#### WINSTATIONCONFIG2

The **WINSTATIONCONFIG2** structure specifies configuration of a session that the user can connect to.

1. typedef struct \_WINSTATIONCONFIG2 {
2. WINSTATIONCREATE Create;
3. PDCONFIG Pd[MAX\_PDCONFIG];
4. WDCONFIG Wd;
5. CDCONFIG Cd;
6. WINSTATIONCONFIG Config;
7. } WINSTATIONCONFIG2, \* PWINSTATIONCONFIG2;

**Create:**  General creation information.

**Pd:**  An array of protocol data configuration structures for this [**WinStation**](#gt_8cc89bb9-be94-4672-8a5d-9947b91bc952).

**Wd:**  The WinStation (session) driver for this WinStation configuration.

**Cd:**  The connection driver for this WinStation configuration.

**Config:**  The specific configuration values for the WinStation (session).

##### WINSTATIONCONFIG

[**WinStation**](#gt_8cc89bb9-be94-4672-8a5d-9947b91bc952) configuration data. Included inside a [WINSTATIONCONFIG2](#Section_38d998ad51e744358231fff55348fff8) structure.

1. typedef struct \_WINSTATIONCONFIG {
2. WCHAR Comment[WINSTATIONCOMMENT\_LENGTH + 1];
3. USERCONFIG User;
4. char OEMId[4];
5. } WINSTATIONCONFIG,
6. \*PWINSTATIONCONFIG;

**Comment:**  The WinStation descriptive comment.

**User:**  The user configuration data for the session (WinStation).

**OEMId:**  Value identifying who implemented the TermService Listener that this session (WinStation) belongs to. This can be any value defined by the implementer (OEM) of the [**listener**](#gt_1e5aa0c2-4617-4d7a-969e-a7091c415c8d).

#### POLICY\_TS\_MACHINE

The **POLICY\_TS\_MACHINE** structure defines the machine policy of the server. Each item in the policy has a flag to indicate if the policy is present and a value for the policy.[<105>](#Appendix_A_105" \o "Product behavior note 105)

1. typedef struct \_POLICY\_TS\_MACHINE {
2. ULONG fPolicyDisableClip  :1;
3. ULONG fPolicyDisableCam  :1;
4. ULONG fPolicyDisableCcm  :1;
5. ULONG fPolicyDisableLPT  :1;
6. ULONG fPolicyDisableCpm  :1;
7. ULONG fPolicyPromptForPassword  :1;
8. ULONG fPolicyMaxInstanceCount  :1;
9. ULONG fPolicyMinEncryptionLevel  :1;
10. ULONG fPolicyFipsEnabled  :1;
11. ULONG fPolicyDisableAutoReconnect  :1;
12. ULONG fPolicyWFProfilePath  :1;
13. ULONG fPolicyWFHomeDir  :1;
14. ULONG fPolicyWFHomeDirDrive  :1;
15. ULONG fPolicyDenyTSConnections  :1;
16. ULONG fPolicyTempFoldersPerSession  :1;
17. ULONG fPolicyDeleteTempFoldersOnExit  :1;
18. ULONG fPolicyColorDepth  :1;
19. ULONG fPolicySessionDirectoryActive  :1;
20. ULONG fPolicySessionDirectoryLocation  :1;
21. ULONG fPolicySessionDirectoryClusterName  :1;
22. ULONG fPolicySessionDirectoryAdditionalParams  :1;
23. ULONG fPolicySessionDirectoryExposeServerIP  :1;
24. ULONG fPolicyPreventLicenseUpgrade  :1;
25. ULONG fPolicySecureLicensing  :1;
26. ULONG fPolicyWritableTSCCPermissionsTAB  :1;
27. ULONG fPolicyDisableCdm  :1;
28. ULONG fPolicyForceClientLptDef  :1;
29. ULONG fPolicyShadow  :1;
30. ULONG fPolicyResetBroken  :1;
31. ULONG fPolicyReconnectSame  :1;
32. ULONG fPolicyMaxSessionTime  :1;
33. ULONG fPolicyMaxDisconnectionTime  :1;
34. ULONG fPolicyMaxIdleTime  :1;
35. ULONG fPolicyInitialProgram  :1;
36. ULONG fPolicySingleSessionPerUser  :1;
37. ULONG fPolicyDisableWallpaper  :1;
38. ULONG fPolicyKeepAlive  :1;
39. ULONG fPolicyEnableTimeZoneRedirection  :1;
40. ULONG fPolicyDisableForcibleLogoff  :1;
41. ULONG fPolicyLicensingMode  :1;
42. ULONG fPolicyExplicitLSDiscovery  :1;
43. ULONG fPolicyDisableTerminalServerTooltip  :1;
44. ULONG fDisableClip  :1;
45. ULONG fDisableCam  :1;
46. ULONG fDisableCcm  :1;
47. ULONG fDisableLPT  :1;
48. ULONG fDisableCpm  :1;
49. ULONG fPromptForPassword  :1;
50. ULONG ColorDepth  :3;
51. ULONG fDenyTSConnections  :1;
52. ULONG fTempFoldersPerSession  :1;
53. ULONG fDeleteTempFoldersOnExit  :1;
54. ULONG fWritableTSCCPermissionsTAB  :1;
55. ULONG fDisableCdm  :1;
56. ULONG fForceClientLptDef  :1;
57. ULONG fResetBroken  :1;
58. ULONG fReconnectSame  :1;
59. ULONG fSingleSessionPerUser  :1;
60. ULONG fDisableWallpaper  :1;
61. ULONG fKeepAliveEnable  :1;
62. ULONG fPreventLicenseUpgrade  :1;
63. ULONG fSecureLicensing  :1;
64. ULONG fEnableTimeZoneRedirection  :1;
65. ULONG fDisableAutoReconnect  :1;
66. ULONG fDisableForcibleLogoff  :1;
67. ULONG fPolicyEncryptRPCTraffic  :1;
68. ULONG fEncryptRPCTraffic  :1;
69. ULONG fErrorInvalidProfile  :1;
70. ULONG fPolicyFallbackPrintDriver  :1;
71. ULONG FallbackPrintDriverType  :3;
72. ULONG fDisableTerminalServerTooltip  :1;
73. BYTE bSecurityLayer;
74. ULONG fPolicySecurityLayer  :1;
75. BYTE bUserAuthentication;
76. ULONG fPolicyUserAuthentication  :1;
77. ULONG fPolicyTurnOffSingleAppMode  :1;
78. ULONG fTurnOffSingleAppMode  :1;
79. ULONG fDisablePNPPolicyIsEnfored  :1;
80. ULONG fDisablePNPPolicyValue  :1;
81. ULONG MaxInstanceCount;
82. ULONG LicensingMode;
83. BYTE MinEncryptionLevel;
84. WCHAR WFProfilePath[DIRECTORY\_LENGTH + 1];
85. WCHAR WFHomeDir[DIRECTORY\_LENGTH + 1];
86. WCHAR WFHomeDirDrive[ 4 ];
87. ULONG SessionDirectoryActive;
88. WCHAR SessionDirectoryLocation[DIRECTORY\_LENGTH+1];
89. WCHAR SessionDirectoryClusterName[DIRECTORY\_LENGTH+1];
90. WCHAR SessionDirectoryAdditionalParams[DIRECTORY\_LENGTH+1];
91. ULONG SessionDirectoryExposeServerIP;
92. ULONG KeepAliveInterval;
93. SHADOWCLASS Shadow;
94. ULONG MaxConnectionTime;
95. ULONG MaxDisconnectionTime;
96. ULONG MaxIdleTime;
97. WCHAR WorkDirectory[DIRECTORY\_LENGTH+1];
98. WCHAR InitialProgram[INITIALPROGRAM\_LENGTH + 1];
99. WCHAR LicenseServers[MAX\_LICENSE\_SERVER\_LENGTH + 1];
100. } POLICY\_TS\_MACHINE,
101. \*PPOLICY\_TS\_MACHINE;

**fPolicyDisableClip:**  TRUE indicates the policy for DisableClip is set; FALSE otherwise.

**fPolicyDisableCam:**  TRUE indicates the policy for DisableCam is set; FALSE otherwise.

**fPolicyDisableCcm:**  TRUE indicates the policy for DisableCcm is set; FALSE otherwise.

**fPolicyDisableLPT:**  TRUE indicates the policy for DisableLPT is set; FALSE otherwise.

**fPolicyDisableCpm:**  TRUE indicates the policy for DisableCpm is set; FALSE otherwise.

**fPolicyPromptForPassword:**  TRUE indicates the policy for PromptForPassword is set; FALSE otherwise.

**fPolicyMaxInstanceCount:**  TRUE indicates the policy for MaxInstanceCount is set; FALSE otherwise.

**fPolicyMinEncryptionLevel:**  TRUE indicates the policy for MinEncryptionLevel is set; FALSE otherwise.

**fPolicyFipsEnabled:**  TRUE indicates the policy for Fips is enabled; FALSE otherwise.[<106>](#Appendix_A_106" \o "Product behavior note 106)

**fPolicyDisableAutoReconnect:**  TRUE indicates the policy for DisableAutoReconnect is set; FALSE otherwise.

**fPolicyWFProfilePath:**  TRUE indicates the policy for WFProfilePath is set; FALSE otherwise.

**fPolicyWFHomeDir:**  TRUE indicates the policy for WFHomeDir is set; FALSE otherwise.

**fPolicyWFHomeDirDrive:**  TRUE indicates the policy for WFHomeDirDrive is set; FALSE otherwise.

**fPolicyDenyTSConnections:**  TRUE indicates the policy for DenyTSConnections is set; FALSE otherwise.

**fPolicyTempFoldersPerSession:**  TRUE indicates the policy for TempFoldersPerSession is set; FALSE otherwise.

**fPolicyDeleteTempFoldersOnExit:**  TRUE indicates the policy for DeleteTempFoldersOnExit is set; FALSE otherwise.

**fPolicyColorDepth:**  TRUE indicates the policy for ColorDepth is set; FALSE otherwise.

**fPolicySessionDirectoryActive:**  TRUE indicates the policy for SessionDirectoryActive is set; FALSE otherwise.

**fPolicySessionDirectoryLocation:**  TRUE indicates the policy for SessionDirectoryLocation is set; FALSE otherwise.

**fPolicySessionDirectoryClusterName:**  TRUE indicates the policy for SessionDirectoryClusterName is set; FALSE otherwise.

**fPolicySessionDirectoryAdditionalParams:**  TRUE indicates the policy for SessionDirectoryAdditionalParams is set; FALSE otherwise.

**fPolicySessionDirectoryExposeServerIP:**  TRUE indicates the policy for SessionDirectoryExposeServerIP is set; FALSE otherwise.

**fPolicyPreventLicenseUpgrade:**  TRUE indicates the policy for PreventLicenseUpgrade is set; FALSE otherwise.

**fPolicySecureLicensing:**  TRUE indicates the policy for SecureLicensing is set; FALSE otherwise.[<107>](#Appendix_A_107" \o "Product behavior note 107)

**fPolicyWritableTSCCPermissionsTAB:**  TRUE indicates the policy for WritableTSCCPermissionsTAB is set; FALSE otherwise.

**fPolicyDisableCdm:**  TRUE indicates the policy for DisableCdm is set; FALSE otherwise.

**fPolicyForceClientLptDef:**  TRUE indicates the policy for ForceClientLptDef is set; FALSE otherwise.

**fPolicyShadow:**  TRUE indicates the policy for Shadow is set; FALSE otherwise.

**fPolicyResetBroken:**  TRUE indicates the policy for ResetBroken is set; FALSE otherwise.

**fPolicyReconnectSame:**  TRUE indicates the policy for ReconnectSame is set; FALSE otherwise.

**fPolicyMaxSessionTime:**  TRUE indicates the policy for MaxSessionTime is set; FALSE otherwise.

**fPolicyMaxDisconnectionTime:**  TRUE indicates the policy for MaxDisconnectionTime is set; FALSE otherwise.

**fPolicyMaxIdleTime:**  TRUE indicates the policy for MaxIdleTime is set; FALSE otherwise.

**fPolicyInitialProgram:**  TRUE indicates the policy for InitialProgram is set; FALSE otherwise.

**fPolicySingleSessionPerUser:**  TRUE indicates the policy for SingleSessionPerUser is set; FALSE otherwise.

**fPolicyDisableWallpaper:**  TRUE indicates the policy for DisableWallpaper is set; FALSE otherwise.

**fPolicyKeepAlive:**  TRUE indicates the policy for KeepAlive is set; FALSE otherwise.

**fPolicyEnableTimeZoneRedirection:**  TRUE indicates the policy for EnableTimeZoneRedirection is set; FALSE otherwise.[<108>](#Appendix_A_108" \o "Product behavior note 108)

**fPolicyDisableForcibleLogoff:**  TRUE indicates the policy for DisableForcibleLogoff is set; FALSE otherwise.[<109>](#Appendix_A_109" \o "Product behavior note 109)

**fPolicyLicensingMode:**  TRUE indicates the policy for LicensingMode is set; FALSE otherwise.[<110>](#Appendix_A_110" \o "Product behavior note 110)

**fPolicyExplicitLSDiscovery:**  TRUE indicates the policy for ExplicitLSDiscovery is set; FALSE otherwise.[<111>](#Appendix_A_111" \o "Product behavior note 111)

**fPolicyDisableTerminalServerTooltip:**  TRUE indicates the policy for DisableTerminalServerTooltip is set; FALSE otherwise.[<112>](#Appendix_A_112" \o "Product behavior note 112)

**fDisableClip:**  TRUE indicates disable client clipboard redirection; FALSE otherwise.

**fDisableCam:**  TRUE indicates disable client audio redirection; FALSE otherwise.

**fDisableCcm:**  TRUE indicates disable client COM port redirection; FALSE otherwise.

**fDisableLPT:**  TRUE indicates disable client LPT port redirection; FALSE otherwise.

**fDisableCpm:**  TRUE indicates disable client printer redirection; FALSE otherwise.

**fPromptForPassword:**  Set to FALSE to log on user with previously provided credentials, or TRUE to prompt the user for password.

**ColorDepth:** The color depth of the session.[<113>](#Appendix_A_113" \o "Product behavior note 113) The following supported values translate to the number of colors supported:

* 0x1 256 (8 bpp)
* 0x2 32,768 (15 bpp)
* 0x3 65,536 (16 bpp)
* 0x4 16 million (24 bpp)
* 0x5 16 million with transparency (32 bpp)

**fDenyTSConnections:**  If set to TRUE, [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) is effectively disabled since remote connections will be declined; FALSE otherwise.

**fTempFoldersPerSession:**  Set to TRUE if there are temporary folders per session instead of one common temp folder, FALSE otherwise.

**fDeleteTempFoldersOnExit:**  If set to TRUE, delete temporary folders on session exit; FALSE otherwise.

**fWritableTSCCPermissionsTAB:**  If set to TRUE, an [**administrator**](#gt_eb335f5b-619b-46ed-9138-06e9910108c3) can change the per-connection security description, FALSE otherwise.

**fDisableCdm:**  TRUE indicates disable client drive redirection; FALSE otherwise.

**fForceClientLptDef:**  TRUE indicates force the client's redirected printer to be the default printer for the user; FALSE otherwise.

**fResetBroken:**  TRUE indicates reset the session if the connection is broken or if the connection or idle timers expire; FALSE otherwise.

**fReconnectSame:**  Set to FALSE to indicate that the user can reconnect from any client computer to a disconnected session. TRUE indicates that the user can reconnect to a disconnected session only from the same client computer that initially established the disconnected session. Logging on from a different client computer will lead to a new Terminal Services session being created.

**fSingleSessionPerUser:**  TRUE indicates each user can have only a single session; FALSE otherwise.

**fDisableWallpaper:**  TRUE indicates display of the desktop wallpaper in the session has been disabled; FALSE otherwise.

**fKeepAliveEnable:**  TRUE indicates KeepAlive is enabled; FALSE otherwise.

**fPreventLicenseUpgrade:**  TRUE indicates licenses are prevented from being upgraded; FALSE otherwise.

**fSecureLicensing:**  TRUE indicates secure licensing is enabled; FALSE otherwise.[<114>](#Appendix_A_114" \o "Product behavior note 114)

**fEnableTimeZoneRedirection:**  TRUE indicates Client time zone redirection is enabled; FALSE otherwise.[<115>](#Appendix_A_115" \o "Product behavior note 115)

**fDisableAutoReconnect:**  TRUE indicates disable auto-reconnect functionality; FALSE otherwise.

**fDisableForcibleLogoff:**  TRUE indicates disable forcible logoff; FALSE otherwise.[<116>](#Appendix_A_116" \o "Product behavior note 116)

**fPolicyEncryptRPCTraffic:**  TRUE indicates policy for EncryptRpcTraffic is set; FALSE otherwise.[<117>](#Appendix_A_117" \o "Product behavior note 117)

**fEncryptRPCTraffic:**  TRUE indicates the policy for EncryptRpcTraffic is set; FALSE otherwise.

**fErrorInvalidProfile:**  Set to TRUE if WFProfilePath, WFHomeDir, or WFHomeDirDrive is invalid (too long), FALSE otherwise.[<118>](#Appendix_A_118" \o "Product behavior note 118)

**fPolicyFallbackPrintDriver:**  TRUE indicates the policy for FallbackPrintDriver is set; FALSE otherwise.[<119>](#Appendix_A_119" \o "Product behavior note 119)

**FallbackPrintDriverType:**  The fallback printer driver type. Can be any of the following values:[<120>](#Appendix_A_120" \o "Product behavior note 120)

• NO\_FALLBACK\_DRIVERS (0x0)

• FALLBACK\_BESTGUESS (0x1)

• FALLBACK\_PCL (0x2)

• FALLBACK\_PS (0x3)

• FALLBACK\_PCLANDPS (0x4)

**fDisableTerminalServerTooltip:**  TRUE indicates disable [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) tooltip; FALSE otherwise.[<121>](#Appendix_A_121" \o "Product behavior note 121)

**bSecurityLayer:**  If non-zero, indicates the SSL security layer in use.[<122>](#Appendix_A_122" \o "Product behavior note 122)

**fPolicySecurityLayer:**  TRUE indicates the policy for SecurityLayer is set; FALSE otherwise.[<123>](#Appendix_A_123" \o "Product behavior note 123)

**bUserAuthentication:**  The user authentication level. It can be any of the following values:

* TS\_USER\_AUTHENTICATION\_NONE
* TS\_USER\_AUTHENTICATION\_VIA\_HYBRID
* TS\_USER\_AUTHENTICATION\_VIA\_SSL
* TS\_USER\_AUTHENTICATION\_DEFAULT (same as TS\_USER\_AUTHENTICATION\_NONE)

**fPolicyUserAuthentication:**  TRUE indicates the policy for UserAuthentication is set; FALSE otherwise.[<124>](#Appendix_A_124" \o "Product behavior note 124)

**fPolicyTurnOffSingleAppMode:**  TRUE indicates the policy for TurnOffSingleAppMode is set; FALSE otherwise.[<125>](#Appendix_A_125" \o "Product behavior note 125)

**fTurnOffSingleAppMode:**  TRUE specifies that the desktop is always displayed when a client connects to a remote computer. FALSE specifies an initial program can be specified that runs on the remote computer after the client connects to the remote computer.

**fDisablePNPPolicyIsEnfored:**  TRUE indicates policy for PnP redirection is set, FALSE otherwise.

**fDisablePNPPolicyValue:**  TRUE indicates disable PnP redirection, FALSE otherwise.

**MaxInstanceCount:**  The maximum number of instances that can connect.

**LicensingMode:**  The licensing mode of the server.

**MinEncryptionLevel:**  The minimum allowed encryption level. Possible numeric values for this parameter include 1 (Low), 2 (Client Compatible), 3 (High), and 4 (FIPS). Detailed description of these encryption levels is included in [[MS-RDPBCGR]](%5bMS-RDPBCGR%5d.pdf#Section_5073f4ed1e9345e1b0396e30c385867c) sections 5.3.1 and 5.4.1.

**WFProfilePath:**  The Terminal Services profile path. Overrides standard profile path.

**WFHomeDir:**  The Terminal Services home directory path. Overrides standard home directory.

**WFHomeDirDrive:**  The Terminal Services home directory drive. Overrides standard home directory.

**SessionDirectoryActive:**  Set to TRUE if the machine is part of a Terminal Server Farm, FALSE otherwise. For information about Terminal Server Farms, see [[MSFT-SDLBTS]](https://go.microsoft.com/fwlink/?LinkId=90204).

**SessionDirectoryLocation:**  The name of the Session Directory Server. For information about Session Directory, see [MSFT-SDLBTS].

**SessionDirectoryClusterName:**  The name of the Terminal Server Farm to which this machine belongs. For information about Terminal Server Farms, see [MSFT-SDLBTS].

**SessionDirectoryAdditionalParams:**  Additional parameters to pass to the session directory. This is an opaque type.

**SessionDirectoryExposeServerIP:**  If set to TRUE, expose the server's IP address to the client; otherwise FALSE.

**KeepAliveInterval:**  Specifies the interval between keep-alives.

**Shadow:**  Specifies whether shadowing of the session is allowed.

**MaxConnectionTime:**  The maximum allowed session connection time setting of the session in milliseconds.

**MaxDisconnectionTime:**  The maximum allowed session disconnect time of the session in milliseconds.

**MaxIdleTime:**  The maximum allowed session idle time setting of the session in milliseconds.

**WorkDirectory:**  The work directory for the initial program.

**InitialProgram:**  The program to run instead of the default, if set.[<126>](#Appendix_A_126" \o "Product behavior note 126)

**LicenseServers:**  A hardcoded array of license servers that the server will use instead of using license server discovery.

#### WINSTATIONUSERTOKEN

The WINSTATIONUSERTOKEN structure defines the user token for a session.

1. typedef struct \_WINSTATIONUSERTOKEN {
2. HANDLE ProcessId;
3. HANDLE ThreadId;
4. HANDLE UserToken;
5. } WINSTATIONUSERTOKEN,
6. \*PWINSTATIONUSERTOKEN;

**ProcessId:**  Specifies the Process ID.

**ThreadId:**  Specifies the [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to the calling thread.

**UserToken:**  Returns the user token that is currently logged on to the session.

#### WINSTATIONVIDEODATA

The WINSTATIONVIDEODATA structure defines the resolution and color depth of a session.

1. typedef struct \_WINSTATIONVIDEODATA {
2. USHORT HResolution;
3. USHORT VResolution;
4. USHORT fColorDepth;
5. } WINSTATIONVIDEODATA,
6. \*PWINSTATIONVIDEODATA;

**HResolution:**  Specifies the horizontal resolution, in pixels.

**VResolution:**  Specifies the vertical resolution, in pixels.

**fColorDepth:**  Specifies the color depth. The supported values 1, 2, 4, 8, and 16 are translated, respectively, as the following number of colors supported: 16 (4 bpp), 256 (8 bpp), 65,536 (16 bpp), 16 million (24 bpp), 32,768 (15 bpp).[<127>](#Appendix_A_127" \o "Product behavior note 127)

#### WINSTATIONLOADINDICATORDATA

The WINSTATIONLOADINDICATORDATA structure defines data used for the load balancing of a server.

1. typedef struct \_WINSTATIONLOADINDICATORDATA {
2. ULONG RemainingSessionCapacity;
3. LOADFACTORTYPE LoadFactor;
4. ULONG TotalSessions;
5. ULONG DisconnectedSessions;
6. LARGE\_INTEGER IdleCPU;
7. LARGE\_INTEGER TotalCPU;
8. ULONG RawSessionCapacity;
9. ULONG reserved[9];
10. } WINSTATIONLOADINDICATORDATA,
11. \*PWINSTATIONLOADINDICATORDATA;

**RemainingSessionCapacity:**  The estimated number of additional sessions that can be supported given the CPU constraint.

**LoadFactor:**  Indicates the most constrained current resource.

**TotalSessions:**  The total number of sessions.

**DisconnectedSessions:**  The number of disconnected sessions.

**IdleCPU:**  This is always set to 0.

**TotalCPU:**  This is always set to 0.

**RawSessionCapacity:**  The raw number of sessions capacity.

**reserved:**  Reserved.

##### LOADFACTORTYPE

The LOADFACTORTYPE enumeration specifies the most constrained resource affecting load balancing.

1. typedef enum \_LOADFACTORTYPE
2. {
3. ErrorConstraint,
4. PagedPoolConstraint,
5. NonPagedPoolConstraint,
6. AvailablePagesConstraint,
7. SystemPtesConstraint,
8. CPUConstraint
9. } LOADFACTORTYPE;

**ErrorConstraint:** An error occurred while obtaining constraint data.

**PagedPoolConstraint:** The amount of paged pool is the constraint.

**NonPagedPoolConstraint:** The amount of non-paged pool is the constraint.

**AvailablePagesConstraint:** The amount of available pages is the constraint.

**SystemPtesConstraint:** The number of system page table entries (PTEs) is the constraint.

**CPUConstraint:** CPU usage is the constraint.

#### WINSTATIONSHADOW

The WINSTATIONSHADOW structure is used for [RpcWinStationQueryInformation](#Section_1bba9ff271d349a3bb262e5f6fcab3ee) and [RpcWinStationSetInformation](#Section_2a5ee131a1dd44c7988098df708061ea) operations.

1. typedef struct \_WINSTATIONSHADOW {
2. SHADOWSTATECLASS ShadowState;
3. SHADOWCLASS ShadowClass;
4. ULONG SessionId;
5. ULONG ProtocolType;
6. } WINSTATIONSHADOW,
7. \*PWINSTATIONSHADOW;

**ShadowState:**  Specifies the current state of shadowing.

**ShadowClass:**  Specifies the type of shadowing.

**SessionId:**  Specifies the session ID of the session.

**ProtocolType:**  Specifies the type of protocol on the session. Can be one of the following values.

| Name | Value |
| --- | --- |
| PROTOCOL\_OTHERS | 1 |
| PROTOCOL\_RDP | 2 |
| PROTOCOL\_CONSOLE | 0[<128>](#Appendix_A_128" \o "Product behavior note 128) |

##### SHADOWSTATECLASS

The **SHADOWSTATECLASS** enumeration specifies [**WinStation**](#gt_8cc89bb9-be94-4672-8a5d-9947b91bc952) shadow states.

1. typedef enum \_SHADOWSTATECLASS
2. {
3. State\_NoShadow,
4. State\_Shadowing,
5. State\_Shadowed,
6. } SHADOWSTATECLASS;

**State\_NoShadow:** No shadow operations are currently being performed on this session.

**State\_Shadowing:** The session is shadowing a different session. The current session is referred to as a shadow client.

**State\_Shadowed:** The session is being shadowed by a different session. The current session is referred to as a shadow target.

#### WINSTATIONPRODID

The **WINSTATIONPRODID** structure represents a product ID for the session.

1. typedef struct \_WINSTATIONPRODID {
2. WCHAR DigProductId[CLIENT\_PRODUCT\_ID\_LENGTH];
3. WCHAR ClientDigProductId[CLIENT\_PRODUCT\_ID\_LENGTH];
4. WCHAR OuterMostDigProductId[CLIENT\_PRODUCT\_ID\_LENGTH];
5. ULONG curentSessionId;
6. ULONG ClientSessionId;
7. ULONG OuterMostSessionId;
8. } WINSTATIONPRODID,
9. \*PWINSTATIONPRODID;

**DigProductId:**  The product ID of the server. For information about the ProductID property, see [[MSDN-ProductID]](https://go.microsoft.com/fwlink/?LinkId=207862).[<129>](#Appendix_A_129" \o "Product behavior note 129)

**ClientDigProductId:**  The product ID of the client.[<130>](#Appendix_A_130" \o "Product behavior note 130)

**OuterMostDigProductId:**  Not used.

**curentSessionId:**  The current session identifier.

**ClientSessionId:**  The client's session identifier.

**OuterMostSessionId:**  Not used.

#### WINSTATIONREMOTEADDRESS

The WINSTATIONREMOTEADDRESS structure specifies the client's remote address. Only TCP/IP addresses are supported.[<131>](#Appendix_A_131" \o "Product behavior note 131)

1. typedef struct {
2. unsigned short sin\_family;
3. union {
4. struct {
5. USHORT sin\_port;
6. ULONG in\_addr;
7. UCHAR sin\_zero[8];
8. } ipv4;
9. struct {
10. USHORT sin6\_port;
11. ULONG sin6\_flowinfo;
12. USHORT sin6\_addr[8];
13. ULONG sin6\_scope\_id;
14. } ipv6;
15. };
16. } WINSTATIONREMOTEADDRESS,
17. \*PWINSTATIONREMOTEADDRESS;

**sin\_family:**  MUST be AF\_INET to indicate that IPv4 is supported or AF\_INET6 to indicate that IPv6 is supported. For more information on AF\_INET and AF\_INET6, see [[MSDN-SOCKET]](https://go.microsoft.com/fwlink/?LinkId=93442).

**ipv4:**  IPv4 address. For more information, see [[MSDN-TDIADDRESS]](https://go.microsoft.com/fwlink/?LinkId=93316).

**sin\_port:**  Specifies a TCP or User Datagram Protocol (UDP) port number.

**in\_addr:**  Indicates the IP address.

**sin\_zero:**  An array filled with zeros.

**ipv6:**  IPv6 address.

**sin6\_port:**  Specifies a TCP or UDP port number.

**sin6\_flowinfo:**  Ipv6 flow information.

**sin6\_addr:**  Indicates the IP address.

**sin6\_scope\_id:**  Set of interfaces for a scope. For more information, see [[MSDN-SOCKADDR\_IN6]](https://go.microsoft.com/fwlink/?LinkId=206745).

#### ExtendedClientCredentials

The ExtendedClientCredentials structure holds longer user name, password, and [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) fields.[<132>](#Appendix_A_132" \o "Product behavior note 132)

1. typedef struct \_ExtendedClientCredentials {
2. WCHAR UserName[EXTENDED\_USERNAME\_LEN + 1];
3. WCHAR Password[EXTENDED\_PASSWORD\_LEN + 1];
4. WCHAR Domain[EXTENDED\_DOMAIN\_LEN + 1];
5. } ExtendedClientCredentials,
6. \*pExtendedClientCredentials;

**UserName:**  Specifies the user's username.

**Password:**  Specifies the user's password.

**Domain:**  Specifies the user's domain name.

#### TS\_TRACE

The TS\_TRACE structure specifies fields used for configuring tracing operations in TS binaries if they are checked.

1. typedef struct \_TS\_TRACE {
2. WCHAR TraceFile[256];
3. BOOLEAN fDebugger;
4. BOOLEAN fTimestamp;
5. ULONG TraceClass;
6. ULONG TraceEnable;
7. WCHAR TraceOption[64];
8. } TS\_TRACE,
9. \*PTS\_TRACE;

**TraceFile:**  Specifies the file name, if any, to which to write debug information.

**fDebugger:**  Specifies whether debugger is attached.

**fTimestamp:**  Specifies whether to append time stamp to the traces logged.

**TraceClass:**  Classes of tracing to log. They enable tracing for the various [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) binaries/functionalities. It MUST be a bitwise OR combination of one or more of the following values.

| Value | Meaning |
| --- | --- |
| TC\_ICASRV  0x00000001 | Enable tracing for the [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) Service.[<133>](#Appendix_A_133" \o "Product behavior note 133) |
| TC\_ICAAPI  0x00000002 | Enable tracing for the DLL providing the API for Terminal Services to communicate with the [**WinStation**](#gt_8cc89bb9-be94-4672-8a5d-9947b91bc952) Driver.[<134>](#Appendix_A_134" \o "Product behavior note 134) |
| TC\_ICADD  0x00000004 | Enable tracing for the Terminal Services Device Driver.[<135>](#Appendix_A_135" \o "Product behavior note 135) |
| TC\_WD  0x00000008 | Enable tracing for the WinStation Driver.[<136>](#Appendix_A_136" \o "Product behavior note 136) |
| TC\_CD  0x00000010 | Enable tracing for the Connection Driver.[<137>](#Appendix_A_137" \o "Product behavior note 137) |
| TC\_PD  0x00000020 | Enable tracing for the Protocol Driver.[<138>](#Appendix_A_138" \o "Product behavior note 138) |
| TC\_TD  0x00000040 | Enable tracing for the Transport Driver.[<139>](#Appendix_A_139" \o "Product behavior note 139) |
| TC\_RELIABLE  0x00000100 | Not used. |
| TC\_FRAME  0x00000200 | Enable tracing for the Frame Protocol Driver.[<140>](#Appendix_A_140" \o "Product behavior note 140) |
| TC\_COMP  0x00000400 | Enable tracing for the Compression library.[<141>](#Appendix_A_141" \o "Product behavior note 141) |
| TC\_CRYPT  0x00000800 | Enable tracing for the Encryption binary.[<142>](#Appendix_A_142" \o "Product behavior note 142) |
| TC\_TW  0x10000000 | Not used. |
| TC\_DISPLAY  0x10000000 | Enable tracing for the Display Driver.[<143>](#Appendix_A_143" \o "Product behavior note 143) |
| TC\_WFSHELL  0x20000000 | Not used. |
| TC\_WX  0x40000000 | Enable tracing for the WinStation Extension.[<144>](#Appendix_A_144" \o "Product behavior note 144) |
| TC\_LOAD  0x80000000 | Enable tracing for the Load balancing binary.[<145>](#Appendix_A_145" \o "Product behavior note 145) |
| TC\_ALL  0xffffffff | Everything. |

**TraceEnable:**  Type of tracing calls log. It MUST be a bitwise OR combination of one or more of the following values.

| Value | Meaning |
| --- | --- |
| TT\_API1  0x00000001 | API level 1. |
| TT\_API2  0x00000002 | API level 2. |
| TT\_API3  0x00000004 | API level 3. |
| TT\_API4  0x00000008 | API level 4. |
| TT\_OUT1  0x00000010 | Output level 1. |
| TT\_OUT2  0x00000020 | Output level 2. |
| TT\_OUT3  0x00000040 | Output level 3. |
| TT\_OUT4  0x00000080 | Output level 4. |
| TT\_IN1  0x00000100 | Input level 1. |
| TT\_IN2  0x00000200 | Input level 2. |
| TT\_IN3  0x00000400 | Input level 3. |
| TT\_IN4  0x00000800 | Input level 4. |
| TT\_ORAW  0x00001000 | Raw output data. |
| TT\_IRAW  0x00002000 | Raw input data. |
| TT\_OCOOK  0x00004000 | Cooked output data. |
| TT\_ICOOK  0x00008000 | Cooked input data. |
| TT\_SEM  0x00010000 | Semaphores. |
| TT\_NONE  0x10000000 | Only view errors. |
| TT\_ERROR  0xffffffff | Error condition. |

**TraceOption:**  Trace option string. This SHOULD be in the format "<filename>(start-end)", where <filename> is the name of the file that requires trace to be collected and (start-end) is the starting and ending line numbers during which trace is to be collected. This is an optional parameter and can be an empty string meaning collect trace for all files belonging to TraceClass and all lines in those files.

#### BEEPINPUT

The BEEPINPUT structure performs a beep in the session.

1. typedef struct \_BEEPINPUT {
2. ULONG uType;
3. } BEEPINPUT,
4. \*PBEEPINPUT;

**uType:**  If the session ID is 0, this can be any of the values that can be passed to the standard MessageBeep function ([[MSDN-MSGBeep]](https://go.microsoft.com/fwlink/?LinkId=128440)). If the session ID is not 0, a frequency and duration is chosen by the server to send as a beep to the session.

#### WINSTATIONCLIENTDATA

The **WINSTATIONCLIENTDATA** structure is a ClientData structure used to send data through [RpcWinStationSetInformation](#Section_2a5ee131a1dd44c7988098df708061ea) to the client.

1. typedef struct \_WINSTATIONCLIENTDATA {
2. CLIENTDATANAME DataName;
3. BOOLEAN fUnicodeData;
4. } WINSTATIONCLIENTDATA,
5. \*PWINSTATIONCLIENTDATA;

**DataName:**  Identifies the type of data sent in this WINSTATIONCLIENTDATA structure. The definition is dependent on the caller and on the client receiving it. This MUST be a data name following a format similar to that of the CLIENTDATANAME data type.

**fUnicodeData:**  TRUE indicates data is in [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) format; FALSE otherwise.

#### SESSION\_CHANGE

The **SESSION\_CHANGE** structure contains the ID of a session running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) and a mask of the notifications that were received for that session.

1. typedef struct \_SESSION\_CHANGE {
2. LONG SessionId;
3. TNotificationId NotificationId;
4. } SESSION\_CHANGE,
5. \*PSESSION\_CHANGE;

**SessionId:**  Identifies the session for which notification was received.

**NotificationId:**  Mask of the notifications that were received for this session.

#### RCM\_REMOTEADDRESS

The **RCM\_REMOTEADDRESS** structure defines a remote address.

1. typedef struct {
2. USHORT sin\_family;
3. union switch (USHORT sin\_family) {
4. case 2: struct {
5. USHORT sin\_port;
6. ULONG in\_addr;
7. UCHAR sin\_zero[8];
8. } ipv4;
9. case 23: struct {
10. USHORT sin6\_port;
11. ULONG sin6\_flowinfo;
12. USHORT sin6\_addr[8];
13. ULONG sin6\_scope\_id;
14. } ipv6;
15. };
16. } RCM\_REMOTEADDRESS, \*PRCM\_REMOTEADDRESS;

**sin\_family:**  Specifies the type of IP address. Valid values are 2 for IPv4 addresses, and 23 for IPv6 addresses.

**ipv4:**  IPv4 address. For more information, see [[MSDN-TDIADDRESS]](https://go.microsoft.com/fwlink/?LinkId=93316).

**sin\_port:**  Specifies a TCP or UDP port number.

**in\_addr:**  Indicates the IP address.

**sin\_zero:**  An array filled with zeros.

**ipv6:**  IPv6 address.

**sin6\_port:**  Specifies a TCP or UDP port number.

**sin6\_flowinfo:**  IPv6 flow information.

**sin6\_addr:**  Indicates the IP address.

**sin6\_scope\_id:**  Set of interfaces for a scope. For more information about these interfaces, see [[MSDN-SOCKADDR\_IN6]](https://go.microsoft.com/fwlink/?LinkId=206745).

#### CLIENT\_STACK\_ADDRESS

The **CLIENT\_STACK\_ADDRESS** structure represents the client network address.[<146>](#Appendix_A_146" \o "Product behavior note 146)

1. typedef struct \_CLIENT\_STACK\_ADDRESS {
2. BYTE Address[STACK\_ADDRESS\_LENGTH];
3. } CLIENT\_STACK\_ADDRESS,
4. \*PCLIENT\_STACK\_ADDRESS;

**Address:**  The first two bytes represent the address family to which the client network address belongs. For more information, see [[MSDN-SOCKET]](https://go.microsoft.com/fwlink/?LinkId=93442). The remaining bytes represent the client network address in a TDI\_ADDRESS\_IP structure. For more information, see [[MSDN-TDIADDRESS]](https://go.microsoft.com/fwlink/?LinkId=93316).

#### VARDATA\_WIRE

The VARDATA\_WIRE structure defines the size and offset of the variable-length data succeeding it. This structure is used before variable-length data fields that are returned by using specific WinStationInformationClass classes (see section [3.7.4.1.6](#Section_1bba9ff271d349a3bb262e5f6fcab3ee)).

1. typedef struct \_VARDATA\_WIRE {
2. USHORT Size;
3. USHORT Offset;
4. } VARDATA\_WIRE,
5. \*PVARDATA\_WIRE;

**Size:**  Size of the variable length data, in bytes, succeeding this structure.

**Offset:**  Offset, in bytes, of the succeeding variable-length structure in the whole data BLOB.

#### PDPARAMSWIRE

The PDPARAMSWIRE structure precedes a [PDPARAMS](#Section_ff0e2998fb8b4dffab648427fad556eb) structure and defines the size and offset of the PDPARAMS structures in the complete data BLOB.

1. typedef struct \_PDPARAMSWIRE {
2. SDCLASS SdClass;
3. VARDATA\_WIRE SdClassSpecific;
4. } PDPARAMSWIRE,
5. \*PPDPARAMSWIRE;

**SdClass:**  Value of [SDCLASS](#Section_7de560c9ca6b4741a448c67de0e52e0e) that the succeeding PDPARAMS structure corresponds to.

**SdClassSpecific:**  [VARDATA\_WIRE](#Section_cc15eeb5f98148d193f6d4f9bfdb07ca) structure defining the size and offset of the variable-length PDPARAMS data succeeding it.

#### WINSTACONFIGWIRE

The **WINSTACONFIGWIRE** structure precedes a variable-length user configuration data BLOB and defines the size and offset of the user configuration data.

1. typedef struct \_WINSTACONFIGWIRE {
2. WCHAR Comment[61];
3. char OEMId[4];
4. VARDATA\_WIRE UserConfig;
5. VARDATA\_WIRE NewFields;
6. } WINSTACONFIGWIRE,
7. \*PWINSTACONFIGWIRE;

**Comment:**  The [**WinStation**](#gt_8cc89bb9-be94-4672-8a5d-9947b91bc952) descriptive comment.

**OEMId:**  Value identifying the OEM implementor of the TermService Listener to which this session (WinStation) belongs. This can be any value defined by the implementer (OEM) of the [**listener**](#gt_1e5aa0c2-4617-4d7a-969e-a7091c415c8d).

**UserConfig:**  [VARDATA\_WIRE](#Section_cc15eeb5f98148d193f6d4f9bfdb07ca) structure defining the size and offset of the variable-length user configuration data succeeding it.

**NewFields:**  VARDATA\_WIRE structure defining the size and offset of the variable-length new data succeeding it. This field is not used and is a placeholder for any new data, if and when added.

#### TSVIP\_SOCKADDR

The **TSVIP\_SOCKADDR** structure defines a socket address.

1. typedef struct \_TSVIP\_SOCKADDR {
2. #ifdef \_\_midl
3. union switch (unsigned short sin\_family) u
4. {
5. case 2: // AF\_INET
6. struct {
7. USHORT sin\_port;
8. ULONG in\_addr;
9. UCHAR sin\_zero[8];
10. } ipv4;
11. case 23: // AF\_INET6
12. struct {
13. USHORT sin6\_port;
14. ULONG sin6\_flowinfo;
15. USHORT sin6\_addr[8];
16. ULONG sin6\_scope\_id;
17. } ipv6;
18. };
19. #else
20. USHORT sin\_family;
21. union
22. {
23. struct {
24. USHORT sin\_port;
25. ULONG in\_addr;
26. UCHAR sin\_zero[8];
27. } ipv4;
28. struct {
29. USHORT sin6\_port;
30. ULONG sin6\_flowinfo;
31. USHORT sin6\_addr[8];
32. ULONG sin6\_scope\_id;
33. }ipv6;
34. } u;
35. #endif
36. } TSVIP\_SOCKADDR,
37. \*PTSVIP\_SOCKADDR;

**ipv4:**  IPv4 address. For more information, see [[MSDN-TDIADDRESS]](https://go.microsoft.com/fwlink/?LinkId=93316).

**sin\_port:**   Specifies a TCP or UDP port number.

**in\_addr:**  Indicates the IP address.

**sin\_zero:**  An array filled with zeros.

**ipv6:**  IPv6 flow information.

**sin6\_port:**  Specifies a TCP or UDP port number.

**sin6\_flowinfo:**  IPv6 flow information.

**sin6\_addr:**   Indicates the IP address.

**sin6\_scope\_id:**   Set of interfaces for a scope. For more information, see [[MSDN-SOCKADDR\_IN6]](https://go.microsoft.com/fwlink/?LinkId=206745).

#### TSVIPAddress

The **TSVIPAddress** structure defines a session's IP address.

1. typedef struct \_TSVIPAddress {
2. DWORD dwVersion; //Structure version
3. TSVIP\_SOCKADDR IPAddress; //IPv4 is in network byte order.
4. ULONG PrefixOrSubnetMask; //IPv4 is a mask in network byte order,
5. #ifdef \_\_midl //IPv6 is prefix length.
6. [range(0, TSVIP\_MAX\_ADAPTER\_ADDRESS\_LENGTH)]
7. UINT PhysicalAddressLength;
8. [length\_is(PhysicalAddressLength)]
9. BYTE PhysicalAddress[TSVIP\_MAX\_ADAPTER\_ADDRESS\_LENGTH];
10. #else
11. UINT PhysicalAddressLength;
12. BYTE PhysicalAddress[TSVIP\_MAX\_ADAPTER\_ADDRESS\_LENGTH];
13. #endif
14. ULONG LeaseExpires;
15. ULONG T1;
16. ULONG T2;
17. } TSVIPAddress,
18. \*PTSVIPAddress;

**dwVersion:**  Specifies the current TSVIPAddress structure version. This field MUST be set to 0x01, the only supported version.

**IPAddress:**  Specifies the IP address.

**PrefixOrSubnetMask:**  Subnet mask of the IP address.

**PhysicalAddressLength:**  Number of bytes in the **PhysicalAddress**.

Where TSVIP\_MAX\_ADAPTER\_ADDRESS\_LENGTH is defined as

1. #define TSVIP\_MAX\_ADAPTER\_ADDRESS\_LENGTH 16

**PhysicalAddress:**  The MAC address used to acquire the IP address.

**LeaseExpires:**  The lease expiration time for the IP address.

**T1:**  The time at which a request to renew the IP address will be made.

**T2:**  Not used.

#### TSVIPSession

The TSVIPSession structure defines a session with its IP address information.

1. typedef struct \_TSVIPSession {
2. DWORD dwVersion;
3. DWORD SessionId;
4. TSVIPAddress SessionIP;
5. } TSVIPSession,
6. \*PTSVIPSession;

**dwVersion:**  Specifies the version of the structure. This field MUST be set to 0x01, the only supported version.

**SessionId:**  Specifies the ID of the session.

**SessionIP:**  Specifies the IP address for the session. This is of type [TSVIPAddress](#Section_eb4b65abdbfe481db85f7931ee38c2cc).

#### WINSTATIONVALIDATIONINFORMATION

The WINSTATIONVALIDATIONINFORMATION structure contains information that is required to identify and validate a per-device terminal server (TS) [**client access license (CAL)**](#gt_9b06b9f2-1680-41c7-b89c-e08f9fe92a25) associated with the session.

A TS CAL is a license that is issued to a user or device to allow remote access to sessions on a terminal server. A per-device TS CAL is a license that is issued to a specific client device. For more information, see [[MSFT-WSTSL]](https://go.microsoft.com/fwlink/?LinkId=93371).

1. typedef struct \_WINSTATIONVALIDATIONINFORMATION {
2. WINSTATIONPRODUCTINFO ProductInfo;
3. BYTE License[VALIDATIONINFORMATION\_LICENSE\_LENGTH];
4. ULONG LicenseLength;
5. BYTE HardwareID[VALIDATIONINFORMATION\_HARDWAREID\_LENGTH];
6. ULONG HardwareIDLength;
7. } WINSTATIONVALIDATIONINFORMATION,
8. \*PWINSTATIONVALIDATIONINFORMATION;

**ProductInfo:**  Specifies information that identifies the type of license.

**License:**  The per-device license BLOB associated with the session.

**LicenseLength:**  The length, in bytes, of **License**.

**HardwareID:**  An identifier that uniquely identifies the client device.

**HardwareIDLength:**  The length, in bytes, of **HardwareID**.

#### WINSTATIONPRODUCTINFO

The **WINSTATIONPRODUCTINFO** structure defines the type of license.

1. typedef struct \_WINSTATIONPRODUCTINFO {
2. WCHAR CompanyName[PRODUCTINFO\_COMPANYNAME\_LENGTH];
3. WCHAR ProductID[PRODUCTINFO\_PRODUCTID\_LENGTH];
4. } WINSTATIONPRODUCTINFO,
5. \*PWINSTATIONPRODUCTINFO;

**CompanyName:**  The name associated with the license. All licenses are associated with "Microsoft Corporation".

**ProductID:**  An ID identifying the type of license. The product ID for per-device licenses is "A02".

## Directory Service Schema Elements

The Terminal Services Terminal Server Runtime Interface accesses the following [**directory service**](#gt_c36db657-3138-4d9a-9289-ded5cbb8b40e) attributes in the **user** class listed in the following table.

For the syntactic specifications of the following attributes, see Active Directory Domain Services (AD DS) ([[MS-ADA1]](%5bMS-ADA1%5d.pdf#Section_19528560f41e4623a406dabcfff0660f), [[MS-ADA2]](%5bMS-ADA2%5d.pdf#Section_e20ebc4e528540bab3bdffcb81c2783e), [[MS-ADA3]](%5bMS-ADA3%5d.pdf#Section_4517e8353ee644d4bb95a94b6966bfb0), and [[MS-ADSC]](%5bMS-ADSC%5d.pdf#Section_9abb5e97123d4da99557b353ab79b830)).

| Directory service attributes | Description |
| --- | --- |
| msTSAllowLogon | Not used by Terminal Services. |
| msTSBrokenConnectionAction | Not used by Terminal Services. |
| msTSConnectClientDrives | Not used by Terminal Services. |
| msTSConnectPrinterDrives | Not used by Terminal Services. |
| msTSDefaultToMainPrinter | Not used by Terminal Services. |
| msTSHomeDirectory | Not used by Terminal Services; CtxWFHomeDir defined in section [2.3.2](#Section_0bd1a91d3cd64e97b5fa5cf35d90f005) is used instead. |
| msTSHomeDrive | Not used by Terminal Services; CtxWFHomeDirDrive defined in section 2.3.2 is used instead. |
| msTSInitialProgram | Not used by Terminal Services; CtxInitialProgram defined in section 2.3.2 is used instead. |
| msTSMaxConnectionTime | Not used by Terminal Services; CtxMaxConnectionTime defined in section 2.3.2 is used instead. |
| msTSMaxDisconnectionTime | Not used by Terminal Services; CtxMaxDisconnectionTime defined in section 2.3.2 is used instead. |
| msTSMaxIdleTime | Not used by Terminal Services; CtxMaxIdleTime defined in section 2.3.2 is used instead. |
| msTSProfilePath | Not used by Terminal Services; CtxWFProfilePath defined in section 2.3.2 is used instead. |
| msTSReconnectionAction | Not used by Terminal Services. |
| msTSRemoteControl | Not used by Terminal Services; CtxShadow defined in section 2.3.2 is used instead. |
| msTSWorkDirectory | Not used by Terminal Services; CtxWorkDirectory defined in section 2.3.2 is used instead. |
| userParameters | This attribute contains a binary BLOB composed of various fields of the [USERCONFIG](#Section_dba750b8cb354e889811e2a1f8a10701) structure returned by RpcGetConfigData. For the binary BLOB structure, see [UserParameters (section 2.3.1)](#Section_15f8b01d900d4f89a3fa1681e55f383d). |
| msTSProperty01 | This attribute contains multi-strings that represent the personal desktop assigned to the user. For the structure of this attribute, see [msTSProperty01](#Section_8c87e949691c4083b945fdf12e21585f) (section 2.3.4). |

**Note**  Any of the previously defined properties can be set for either a user or a machine. Machine properties MAY override user properties and hence the value returned by the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) calls as previously defined will depend on whether machine properties are set in addition to the user properties.

### userParameters

The **userParameters** attribute is used by Terminal Services to store the configuration data associated with the user connected to a session running on a terminal server. This configuration data is returned in a [USERCONFIG](#Section_dba750b8cb354e889811e2a1f8a10701) structure by the [RpcGetConfigData](#Section_6c4dc21eea534785b50593d23604f2c8) method. Terminal Services does not use UTF-8 or UTF-16 encoding to store the configuration data in the **userParameters** attribute. Terminal Services stores the user configuration data in the **userParameters** attribute in the following format:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| ReservedData (96 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Signature | | | | | | | | | | | | | | | | TSPropertyCount | | | | | | | | | | | | | | | |
| TSPropertyArray (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**ReservedData (96 bytes):** A 96-byte array of reserved data.[<147>](#Appendix_A_147" \o "Product behavior note 147)

**Signature (2 bytes):** A 2-byte Unicode character. This field is used by [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) to assert the validity of the **TSPropertyCount** and **TSPropertyArray** fields. Terminal Services compares the data contained in this field with Unicode character "P". If this field contains the aforementioned character, then the information inside the **TSPropertyArray** and **TSPropertyCount** fields is considered valid. If it contains a different value, then all information inside the **TSPropertyCount** and **TSPropertyArray** fields is considered invalid.

**TSPropertyCount (2 bytes):** A 2-byte unsigned integer indicating the number of elements in **TSPropertyArray**.

**TSPropertyArray (variable):** A variable-length array of [TSProperty](#Section_0bd1a91d3cd64e97b5fa5cf35d90f005) structures. The number of elements in this array is specified by the field **TSPropertyCount**.

### TSProperty

Following is the format of each TSProperty structure:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 2  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 3  0 | 1 |
| NameLength | | | | | | | | | | | | | | | | ValueLength | | | | | | | | | | | | | | | |
| Type | | | | | | | | | | | | | | | | PropName (variable) | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| PropValue (variable) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**NameLength (2 bytes):** A 2-byte unsigned integer indicating the length of the PropName field in bytes.

**ValueLength (2 bytes):** A 2-byte unsigned integer indicating the length of the PropValue field in bytes.

**Type (2 bytes):** A 2-byte unsigned integer indicating the type of the PropValue field. It can have the following values:

| Value | Meaning |
| --- | --- |
| PROP\_TYPE\_ITEM  0x01 | Indicates that the property contains one item. |

**PropName (variable):** A Unicode string whose length is indicated by the NameLength field. The string indicates the name of the property.

**PropValue (variable):** An encoded binary BLOB of length indicated by the ValueLength field. This field either contains a 32-bit unsigned integer or an array of ASCII characters. See [Encoding and decoding PropValue field](#Section_27cd734ab78041e6b9d5b6c9fba6c16a) for encoding and decoding this BLOB.

The following table describes various PropName fields and associated PropValue fields used by Terminal Services.

| PropName | PropValue type | Description | [MS-TSTS] RPC call that can be used to query this value |
| --- | --- | --- | --- |
| CtxCfgPresent | 32-bit unsigned integer | It indicates presence of valid TSProperty structures in TSPropertyArray. If the TSPropertyArray does not contain a TSProperty structure containing this PropName and PropValue, Terminal Services ignore the remaining contents of TSPropertyArray. The PropValue field can only contain value 0xB00B1E55. | None. |
| CtxCfgFlags1 | 32-bit unsigned integer | Each bit in the PropValue maps to a Boolean field of the [USERCONFIG](#Section_dba750b8cb354e889811e2a1f8a10701) structure returned by the RpcGetConfigData method. For details about each bit, see the table of CtxCfgFlags1 values in this section. | RpcGetConfigData. |
| CtxCallBack | 32-bit unsigned integer | The callback class for callback operations. | Returned by RpcGetConfigData method in Callback field of USERCONFIG structure. |
| CtxKeyboardLayout | 32-bit unsigned integer | The keyboard layout (HKL) of the user session. | Returned by the RpcGetConfigData method in the KeyboardLayout field of the USERCONFIG structure. |
| CtxMinEncryptionLevel | 8-bit integer | The minimum allowed encryption level of the user session. | Returned by the RpcGetConfigData method in the MinEncryptionLevel field of the USERCONFIG structure. |
| CtxNWLogonServer | 32-bit unsigned integer | The NetWare logon server name. | Returned by the RpcGetConfigData method in the NWLogonServer field of the USERCONFIG structure. |
| CtxWFHomeDir | Variable-length ASCII character array | This attribute specifies the home directory for the user. Each user on a terminal server has a unique home directory. This ensures that application information is stored separately for each user in a multi-user environment. To set a home directory on the local computer, the implementer specifies a local path; for example, C:\Path. To set a home directory in a network environment, the implementer MUST first set the CtxWFHomeDirDrive property, and then set this property to a Universal Naming Convention (UNC) path. | Returned by the RpcGetConfigData method in the WFHomeDir field of the USERCONFIG structure. |
| CtxWFHomeDirDrive | Variable-length ASCII character array | This attribute specifies a home drive for the user. In a network environment, this property is a string containing a drive specification (a drive letter followed by a colon) to which the UNC path specified in the TerminalServicesCtxWFHomeDir property is mapped. To set a home directory in a network environment, the implementer MUST first set this property, and then set the CtxWFHomeDir property. | Returned by the RpcGetConfigData method in the WFHomeDirDrive field of the USERCONFIG structure. |
| CtxInitialProgram | Variable-length ASCII character array | This attribute specifies the path and file name of the application that the user requires to start automatically when the user logs on to the terminal server. To set an initial application to start when the user logs on, the implementer MUST first set this property, and then set the CtxWorkDirectory property. If the implementer sets only the CtxInitialProgram property, the application starts in the user's session in the default user directory. | Returned by the RpcGetConfigData method in the InitialProgram field of the USERCONFIG structure. |
| CtxMaxConnectionTime | 32-bit unsigned integer | This attribute specifies the maximum duration (in minutes) of the Terminal Services session. After the specified number of minutes has elapsed, the session can be disconnected or terminated. | Returned by the RpcGetConfigData method in the MaxConnectionTime field of the USERCONFIG structure. |
| CtxMaxDisconnectionTime | 32-bit unsigned integer | This attribute specifies the maximum amount of time (in minutes) that a disconnected Terminal Services session remains active on the terminal server. After the specified number of minutes has elapsed, the session is terminated. | Returned by the RpcGetConfigData method in the MaxDisconnectionTime field of the USERCONFIG structure. |
| CtxMaxIdleTime | 32-bit unsigned integer | This attribute specifies the maximum amount of time (in minutes) that the Terminal Services session can remain idle. After the specified number of minutes has elapsed, the session can be disconnected or terminated. | Returned by the RpcGetConfigData method in the MaxIdleTime field of the USERCONFIG structure. |
| CtxWFProfilePath | Variable-length ASCII character array | This attribute specifies a roaming or mandatory profile path to use when the user logs on to the terminal server. The profile path is in the following network path format: \\servername\profiles folder name\username. | Returned by the RpcGetConfigData method in the WFProfilePath field of the USERCONFIG structure. |
| CtxShadow | 32-bit unsigned integer | This attribute specifies whether to allow remote observation or remote control of the user's Terminal Services session. The values are as follows:  0. Disable  1. EnableInputNotify  2. EnableInputNoNotify  3. EnableNoInputNotify  4. EnableNoInputNoNotify  For a description of these values, see [[MSDN-RCMWin32\_TSRCS]](https://go.microsoft.com/fwlink/?LinkId=208353). | Returned by the RpcGetConfigData method in the Shadow field of the USERCONFIG structure. |
| CtxWorkDirectory | Variable-length ASCII character array | This attribute specifies the working directory path for the user. To set an initial application to start when the user logs on to the terminal server, the implementer MUST first set the CtxInitialProgram property, and then set this property. | Returned by the RpcGetConfigData method in the WorkDirectory field of the USERCONFIG structure. |
| CtxCallbackNumber | Variable-length ASCII character array | This attribute specifies the call back number provided to the user on the client side for technical support. | Returned by the RpcGetConfigData method in the CallbackNumber field of the USERCONFIG structure. |

The following table provides the details of each bit in the PropValue associated with the PropName 'CtxCfgFlags1'.

| Bit mask in CtxCfgFlags1 PropValue | [MS-TSTS] RPC call that can be used to query this value |
| --- | --- |
| 0x10000000 | Returned by the RpcGetConfigData method in fInheritInitialProgram field of the USERCONFIG structure. |
| F1MSK\_INHERITCALLBACK  0x08000000 | Returned by the RpcGetConfigData method in the fInheritCallback field of the USERCONFIG structure. |
| F1MSK\_INHERITCALLBACKNUMBER  0x04000000 | Returned by the RpcGetConfigData method in the fInheritCallbackNumber field of the USERCONFIG structure. |
| F1MSK\_INHERITSHADOW  0x02000000 | Returned by the RpcGetConfigData method in the fInheritShadow field of the USERCONFIG structure. |
| F1MSK\_INHERITMAXSESSIONTIME  0x01000000 | Returned by the RpcGetConfigData method in the fInheritMaxSessionTime field of the USERCONFIG structure. |
| F1MSK\_INHERITMAXDISCONNECTIONTIME  0x00800000 | Returned by the RpcGetConfigData method in the fInheritMaxDisconnectionTime field the USERCONFIG structure. |
| F1MSK\_INHERITMAXIDLETIME  0x00400000 | Returned by the RpcGetConfigData method in the fInheritMaxIdleTime field of the USERCONFIG structure. |
| F1MSK\_INHERITAUTOCLIENT  0x00200000 | Returned by the RpcGetConfigData method in the fInheritAutoClient field of the USERCONFIG structure. |
| F1MSK\_INHERITSECURITY  0x00100000 | Returned by the RpcGetConfigData method in the fInheritSecurity field of the USERCONFIG structure. |
| F1MSK\_PROMPTFORPASSWORD  0x00080000 | Returned by the RpcGetConfigData method in the fPromptForPassword field of the USERCONFIG structure. |
| F1MSK\_RESETBROKEN  0x00040000 | Returned by the RpcGetConfigData method in the fResetBroken field of the USERCONFIG structure. |
| F1MSK\_RECONNECTSAME  0x00020000 | Returned by the RpcGetConfigData method in the fReconnectSame field of the USERCONFIG structure. |
| F1MSK\_LOGONDISABLED  0x00010000 | Returned by the RpcGetConfigData method in the fLogonDisabled field of the USERCONFIG structure. |
| F1MSK\_AUTOCLIENTDRIVES  0x00008000 | Returned by the RpcGetConfigData method in the fAutoClientDrives field of the USERCONFIG structure. |
| F1MSK\_AUTOCLIENTLPTS  0x00004000 | Returned by the RpcGetConfigData method in the fAutoClientLpts field of the USERCONFIG structure. |
| F1MSK\_FORCECLIENTLPTDEF  0x00002000 | Returned by the RpcGetConfigData method in the fForceClientLptDef field of the USERCONFIG structure. |
| F1MSK\_DISABLEENCRYPTION  0x00001000 | Returned by the RpcGetConfigData method in the fDisableEncryption field of the USERCONFIG structure. |
| F1MSK\_HOMEDIRECTORYMAPROOT  0x00000800 | Returned by the RpcGetConfigData method in the fHomeDirectoryMapRoot field of the USERCONFIG structure. |
| F1MSK\_USEDEFAULTGINA  0x00000400 | Returned by the RpcGetConfigData method in the fUseDefaultGina field of the USERCONFIG structure. |
| F1MSK\_DISABLECPM  0x00000200 | Returned by the RpcGetConfigData method in the fDisableCpm field of the USERCONFIG structure. |
| F1MSK\_DISABLECDM  0x00000100 | Returned by the RpcGetConfigData method in the fDisableCdm field of the USERCONFIG structure. |
| F1MSK\_DISABLECCM  0x00000080 | Returned by the RpcGetConfigData method in the fDisableCcm field of the USERCONFIG structure. |
| F1MSK\_DISABLELPT  0x00000040 | Returned by the RpcGetConfigData method in the fDisableLPT field of the USERCONFIG structure. |
| F1MSK\_DISABLECLIP  0x00000020 | Returned by the RpcGetConfigData method in the fDisableClip field of the USERCONFIG structure. |
| F1MSK\_DISABLEEXE  0x00000010 | Returned by the RpcGetConfigData method in the fDisableExe field of the USERCONFIG structure. |
| F1MSK\_WALLPAPERDISABLED  0x00000008 | Returned by the RpcGetConfigData method in the fWallPaperDisabled field of the USERCONFIG structure. |
| F1MSK\_DISABLECAM  0x00000004 | Returned by the RpcGetConfigData method in the fDisableCam field of the USERCONFIG structure. |

### Encoding PropValue Field in TSProperty Structure

To create the encoded binary BLOB for the **PropValue** field, for each byte of the input create its [**ASCII**](#gt_79fa85ca-ac61-467c-b819-e97dc1a7a599)-encoded hexadecimal representation and place this representation in 2 consecutive bytes of the output buffer, the most significant hexadecimal digit first followed by the least significant hexadecimal digit. For example, if the input byte contains the ASCII representation of character 'A', the resulting output will be a sequence of two ASCII characters: character '4' followed by character '1' because the hexadecimal representation of a byte that contains the ASCII character 'A' is 41. Hence, the output buffer corresponding to the input buffer byte containing character 'A' will be a sequence of 2 bytes whose hexadecimal representations are 34 and 31. As another example, the input buffer containing the ASCII string "ABCDE\0" would be encoded into the ASCII string "414243444500" (without the terminating 0), which is the same as a sequence of 12 bytes whose hexadecimal representations are 34, 31, 34, 32, 34, 33, 34, 34, 34, 35, 30, and 30.

See [Encoding/Decoding Example (section 4.5)](#Section_5b5aff03b8b74f9eb15fd4cbe3f03d4e) for an example of code that demonstrates encoding and decoding of the PropValue field.

### msTSProperty01

The personal desktop information stored in the msTSProperty01 attribute consists of multiple strings. Each string has the format <Property name>=<Value>. The following property names are stored in this multi-string attribute:

* machine=<FQDN of the machine>
* vmname=<FQDN of the machine>
* vmtype=2
* plugin=vmresource

The machine and the vmname properties are the FQDN of the machine that is assigned to the user as a personal desktop. The vmtype property describes the desktop type, which can have one of the following values:

* MYDESKTOP\_PHYSICAL (0): Reserved for future use.
* MYDESKTOP\_FARM (1): This desktop type means virtual machine belongs to a collection of virtual machines, where a single virtual machine can be assign to the user.
* MYDESKTOP\_NONFARM (2): Reserved for future use.

The plugin property describes the plugin used by the connection broker to process the connection request. The only supported value is "vmresource".

For example, if machine LaBigMac.PulpFiction.com is assigned to user VincentVega, the msTSProperty01 attribute of this user object will have the following strings:

* machine=LaBigMac.PulpFiction.com
* vmname=LaBigMac.PulpFiction.com
* vmtye=2
* plugin=vmresource

# Protocol Details

The methods comprising this [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) interface all return a value greater than or equal to 0 on success and a nonzero, implementation-specific error code on failure. If the return type of the function is HRESULT as specified in section [2.2](#Section_f1a460a6c19e4ad2a1f6d67a71732262), a negative number will be treated as failure. If the return type of the function is Boolean as specified in section 2.2, a nonzero value will be treated as success. Unless otherwise specified in the following sections, a server-side implementation of this protocol can choose any nonzero Win32 error value to signify an error condition, as specified in section [1.8](#Section_0d8eb7e164594729bc72320557ddff32).

The client side of the Terminal Services Terminal Server Runtime Interface Protocol MUST return error codes to the caller without modification. The client side of the Terminal Services Terminal Server Runtime Interface Protocol MUST simply return error codes to the invoking application without taking any protocol action.

Note that the terms "client side" and "server side" refer to the initiating and receiving ends of the protocol, respectively, rather than to client or server versions of an operating system. These methods MUST all behave the same way regardless of whether the "server side" of the protocol is running in a client or server version of an operating system.

## Determining a Caller's Permissions and Access Rights

To determine access rights, the caller's token is retrieved from the [**RPC transport**](#gt_c2eeb200-3cd0-4916-966e-d7d6bff1737a), as described for **RpcImpersonationAccessToken** in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 3.3.3.4.3. It MAY use the [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) that represents the user account in the caller's token. For more information about tokens, see [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.5.2. For more information about SIDs, see [MS-DTYP] section 2.4.2.

### Determining a Caller's Permissions

During processing of methods that implement access checks, this protocol performs access security verification on the caller's identity by using the algorithm specified by the Access Check Algorithm Pseudo code ([[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.5.3.2). The input parameters of that algorithm are mapped as follows:

* **SecurityDescriptor**: This MUST be the **SECURITY\_DESCRIPTOR** of the session.[<148>](#Appendix_A_148" \o "Product behavior note 148) For more information about **SECURITY\_DESCRIPTOR**, see [MS-DTYP] section 2.4.6.
* **Token / Authorization Context**: This MUST be the caller's token.
* **Access Request mask**: This is specified by each method's processing logic and MUST be one or more of the WinStationOpen access values specified in section [6.5](#Section_ce70794f213843e8bf6c2c147887d6a2).
* **Object Tree**: This parameter MUST be NULL.
* **PrincipalSelfSubst SID**: This parameter MUST be NULL.

### Determining Whether a Caller Is SYSTEM

During processing of methods that implement access checks of whether the caller is [**SYSTEM**](#gt_3f2baf2c-3e0b-4764-b7a5-54cebabeb85a), this protocol performs access security verification on the caller's identity by checking whether the [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of the SYSTEM is the same as the SID that represents the user account in the caller's token.

### Determining Whether a Caller Is an Administrator

During processing of methods that implement access checks of whether the caller is an [**administrator**](#gt_eb335f5b-619b-46ed-9138-06e9910108c3), this protocol performs access security verification on the caller's identity by using the support function SidInToken ([[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.5.3.1.1). The input parameters of that algorithm are mapped as follows:

* **Token:** MUST be the caller's token.
* **SidToTest:** MUST be the [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of administrators.
* **PrincipalSelfSubstitute:** MUST be NULL.

### Determining Whether a Caller Is the Same User Who Logged onto the Session

During processing of methods that implement access checks of whether the caller is the same user who logged onto the session, this protocol performs access security verification on the caller's identity by checking whether the [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) of the user logged onto the session is the same as the SID that represents the user account in the caller's token.

## Local Session Manager Client Details

### Abstract Data Model

None.

### Timers

No protocol timers are required beyond those used internally by [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) to implement resiliency to network outages, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15).

### Initialization

The client MUST create an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) connection to the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) by using the details specified in section [2.1](#Section_ac0535d0a8474386ae3ee93a50ff2484).

### Processing Events and Sequencing Rules

When a method completes, the values returned by [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) MUST be returned unmodified to the upper layer.

### Timer Events

No protocol timer events are required on the client except those that are required in the underlying [**RPC transport**](#gt_c2eeb200-3cd0-4916-966e-d7d6bff1737a).

### Other Local Events

None.

## Local Session Manager Server Details

### Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

#### Abstract Data Types

**TS\_COUNTER:** A Terminal Services performance counter structure used to represent a single performance counter. It is described in section [2.2.2.17](#Section_a0173488c4834340ab1aaab2d21ea323). An array of these structures is returned by the [RpcGetSessionCounters](#Section_eeda3f5626de4a6da0737ec5390d7dbd) method.

**dwCounterID:** Identifier of a performance counter. It is used with RpcGetSessionCounters and MUST be set to one of the values described in section [2.2.2.17.1](#Section_e08317cba69340d39a5f61e14cfdb431).

**SessionHandle:** Handle to a session. It is defined in section [2.2.1.1](#Section_08a45954817343c6aa107aeda48e8dc2). The following list shows how **SessionHandle** is used with various methods.

* [RpcOpenSession](#Section_56ede88b913b49119540389fddf879e5) as the *phSession* parameter.
* [RpcCloseSession](#Section_8b79657308b1450e840379ca8fdd80c7) as the *phSession* parameter.
* [RpcConnect](#Section_b57917b9b00648b3ae64016771c25bee) as the *hSession* parameter.
* [RpcDisconnect](#Section_dd882322dcd44c3394e5a1f8e8354ebd) as the *hSession* parameter.
* [RpcLogoff](#Section_6dbdc16149c744deb6aa80283ce07e37) as the *hSession* parameter.
* [RpcGetUserName](#Section_acc0609748ad41a2a0fbfc03e4eba7c5) as the *hSession* parameter.
* [RpcGetTerminalName](#Section_e98ee550643046808ac34bbc25a91a93) as the *hSession* parameter.
* [RpcGetState](#Section_58b06e1dbfe44a2f90a19c9129923e99) as the *hSession* parameter.
* [RpcIsSessionDesktopLocked](#Section_0a12b501a52c41e682b306638cc0ae63) as the *hSession* parameter.
* [RpcShowMessageBox](#Section_63915a11630d4ab686d11034fde1e386) as the *hSession* parameter.
* [RpcGetTimes](#Section_4a7d8496a6c44162adaf7897baa8c4ff) as the *hSession* parameter.

**ExecutionEnvironmentData:** A structure containing information about the [**sessions**](#gt_0cd96b80-a737-4f06-bca4-cf9efb449d12) running on the terminal server and the sessions running on virtual machines hosted on the server. It is used with [RpcGetAllSessions](#Section_68e824e0d78649878ccbc34503fffd3f) and returned as the *ppSessionData* parameter.

**SessionState:** Current state of a session. It MUST be set to one of the values of the **WINSTATIONSTATECLASS** enumeration as defined in section [2.2.1.9](#Section_b619d8aa4e8a4716954ea4d596ed37f8). The following list shows how **SessionState** is used with various methods.

* RpcGetState as the *plState* parameter.
* [RpcGetSessionInformationEx](#Section_7a6b39aa61a34a89b486520e81f4f9e1) as the **SessionState** member of the [LSM\_SESSIONINFO\_EX\_LEVEL1](#Section_2789ef970c7c4ec59f2cd375de4941a0) structure.
* [RpcWaitForSessionState](#Section_637ac10c6570460380223c34ea2ea73a) as the *State* parameter.
* [RpcFilterByState](#Section_14ed1d6871f2449599460116f87ee1ee) as the *State* parameter.
* [RpcGetEnumResultEx](#Section_cb2be08441dd44ec81dc52d371a28554) as the **State** member of the [SESSIONENUM\_LEVEL1](#Section_0c4fde3b4fdb45248cae792d0320247a), [SESSIONENUM\_LEVEL2](#Section_1b889bfe8d324cbabd92f9d7c09d5f60), and [SESSIONENUM\_LEVEL3](#Section_d1662f979e3a4fbbbdd52050df69e902) structures.
* RpcGetAllSessions as the **State** member of the [EXECENVDATA\_LEVEL1](#Section_004305a642654ff5bb8a58fd0f7cf2d5) and [EXECENVDATA\_LEVEL2](#Section_e85daed7ed774c8aafc05596c8d41c9c) structures.

**hEnum:** Handle to the session enumeration object. This is of type [ENUM\_HANDLE](#Section_15f02d59200a435da07b055e1276d01c). It is used with [TermSrvEnumeration](#Section_635a479c7213426fb6beffbe381c6302) methods.

**hNotify:** Handle to the notification object. It is of type [NOTIFY\_HANDLE](#Section_557683a82f4a416abb15fa353fdc4aed). It is used with [TermSrvNotification](#Section_19fec7efb8d040169a823ad091a67372) methods.

**UserSessions:** The number of sessions which are of SESSIONTYPE\_REGULARDESKTOP, or SESSIONTYPE\_ALTERNATESHELL, or SESSIONTYPE\_REMOTEAPP as defined in [SESSIONTYPE (section 2.2.1.18)](#Section_12510151f01d4984a1bb64ec4298c31c). It is used with the [RpcGetLoggedOnCount](#Section_8e02daa60b31494aa3c48c8d73faf442) method and is returned as the *pUserSessions* parameter.

**DeviceSessions:** The number of sessions connected using media center extender device only. These sessions are of SESSIONTYPE\_MEDIACENTEREXT as defined in SESSIONTYPE (section 2.2.1.18). It is used with the RpcGetLoggedOnCount method and is returned as the *pDeviceSessions* parameter. For more information on media center, see [[MSFT-WINMCE]](https://go.microsoft.com/fwlink/?LinkId=208408).

**PSESSIONENUM:** Pointer to a structure containing information about the sessions running on the terminal server. This structure is described in detail in section [2.2.2.4](#Section_fd8759ef4f6c45c6b5429d5e53e90f7d). It is used with the [RpcGetEnumResult](#Section_1a7d5d1d1ce5448fbb4ac79741982edb) method and returned as the *ppSessionEnumResult* parameter.

**SessionInfo:** Union of structures, each structure providing different levels of detail about sessions running on a computer. This union is described in detail in section [2.2.2.4.1](#Section_16c7316fc2ad48c5a5b8d6fd3c2fec36). It is used with the RpcGetEnumResult method and returned as the **Data** field of the *ppSessionEnumResult* parameter.

**SessionInfo\_Ex:** Union of structures, each structure providing different levels of detail about sessions running on a computer. This union is described in detail in section [2.2.2.5.1](#Section_1cae32cd3bd3436db9501f43c2b40863). It is used with the RpcGetEnumResultEx method and returned as the **Data** field of the *ppSessionEnumResult* parameter.

**PSESSIONENUM\_EX:** Pointer to a structure containing information about the sessions running on the terminal server. This structure is described in detail in section [2.2.2.5](#Section_35a722eb00a04ba08e4c735d06378675). It is used with the RpcGetEnumResultEx method and returned as the *ppSessionEnumResult* parameter.

**SESSIONTYPE:** Represents the type of the session as described in section 2.2.1.18. It is used with the [RpcGetSessionType](#Section_e7deb145e33f4ec0a2ac524b0381493d) method and returned as the *pSessionType* parameter.

**ConnectTime:** Represents the most recent time of a connection to the session. It is used with the RpcGetTimes method and returned as the *pConnectTime* parameter.

**DisconnectTime:** Represents the most recent time of a disconnection from the session. It is used with the RpcGetTimes method and returned as the *pDisconnectTime* parameter.

**LogonTime:** Represents the most recent time of a logon to the session. It is used with the RpcGetTimes method and returned as the *pLogonTime* parameter.

**PLSMSESSIONINFORMATION:** Pointer to a structure containing information about a session running on the terminal server. This structure is described in detail in section [2.2.2.8](#Section_fadef65b0d6542b8ad14127106499f85). It is used with the [RpcGetSessionInformation](#Section_1a7fa234480540d3a41c8b55bde2eda0) method and returned as the *pSessionInfo* parameter.

**PLSMSESSIONINFORMATION\_EX:** Pointer to a structure containing information about a session running on the terminal server. This structure is described in detail in section [2.2.2.9](#Section_4e7039f4d1d044f282f008b057476db5). It is used with the RpcGetSessionInformationEx method and returned as the *LSMSessionInfoExPtr* parameter.

**PEXECENVDATA:** Pointer to a structure containing information about the sessions running on the terminal server and the sessions running on virtual machines hosted on the server. This structure is described in detail in section [2.2.2.6](#Section_20bf286ea1584d5fa9f495eeaf1a4869). It is returned by the RpcGetAllSessions method.

**EXECENVDATA\_LEVEL1:** Structure that contains basic information about sessions running on a computer. This structure is described in detail in section 2.2.2.6.1.1. It is used with the RpcGetAllSessions method.

**EXECENVDATA\_LEVEL2:** Structure that contains information about sessions running on a computer that is more detailed than the information contained in **EXECENVDATA\_LEVEL1**. This structure is described in detail in section 2.2.2.6.1.2. It is used with the RpcGetAllSessions method.

**SESSION\_FILTER:** Represents the type of filter to apply when retrieving the list of session IDs running on a terminal server. It is described in section [2.2.2.1](#Section_352695847c5e441081951aa49559274e). It is used with the [RpcGetSessionIds](#Section_922264ed0a2d441fa71c2d3ffa975814) method as the *Filter* parameter.

**SESSION\_CHANGE:** Structure containing the ID of a session running on a terminal server and a mask of the notifications that were received for that session. This structure is described in detail in section [2.2.2.42](#Section_e22785c95a374a47864aed2865d053bc). It is used with the [RpcWaitAsyncNotification](#Section_81d5ba342ae446149aca8deb1eb7a0a1) method.

**PEXECENVDATAEX:** Pointer to a structure containing information about the sessions running on the terminal server and the sessions running on virtual machines hosted on the server. This structure is described in detail in section [2.2.2.7](#Section_0a3f7e7b03f84f62a65fc2d670bb3173). It is returned by the [RpcGetAllSessionsEx](#Section_3f72497925cd41d89829244f4425f9c6) method.

**EXECENVDATAEX\_LEVEL1:** Structure that contains basic information about sessions running on a computer. This structure is described in detail in section [2.2.2.7.1.1](#Section_b4295b0ec64140abbbdbea44323b0804). It is used with the RpcGetAllSessionsEx method.

### Timers

None.

### Initialization

Parameters necessary to initialize the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) protocol are specified in section [2.1](#Section_ac0535d0a8474386ae3ee93a50ff2484).

### Processing Events and Sequencing Rules

This protocol asks the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) runtime to perform a strict Network Data Representation (NDR) data consistency check at target level 6.0 for all methods unless otherwise specified, as defined in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 1.3.

When a method completes, the values returned by RPC MUST be returned unmodified to the upper layer. The methods MAY throw exceptions and the [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) client MUST handle these exceptions by returning the unmodified exception code to the upper layer.

#### TermSrvSession Methods

TermSrvSession provides methods that manage, and provide information about, a session on a given [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). The version for this interface is 1.0.

For information about [**endpoints**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee), [**UUID**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) values, and versions, see sections [2.1](#Section_ac0535d0a8474386ae3ee93a50ff2484) and [1.9](#Section_4f5ff56828874c508b3638bb85170b58).

Methods in RPC Opnum Order

| Method | Description |
| --- | --- |
| [RpcOpenSession](#Section_56ede88b913b49119540389fddf879e5) | Returns a [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to a specified session on the terminal server.  Opnum: 0 |
| [RpcCloseSession](#Section_8b79657308b1450e840379ca8fdd80c7) | Closes the connection to the specified session on the terminal server.  Opnum: 1 |
| [RpcConnect](#Section_b57917b9b00648b3ae64016771c25bee) | Reconnects a session handle returned by RpcOpenSession to another specified session on the terminal server.  Opnum: 2 |
| [RpcDisconnect](#Section_dd882322dcd44c3394e5a1f8e8354ebd) | Disconnects the specified session on the terminal server.  Opnum: 3 |
| [RpcLogoff](#Section_6dbdc16149c744deb6aa80283ce07e37) | Logs off the specified session on the terminal server.  Opnum: 4 |
| [RpcGetUserName](#Section_acc0609748ad41a2a0fbfc03e4eba7c5) | Gets the username and [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) name of the user logged on to the specified session on the terminal server.  Opnum: 5 |
| [RpcGetTerminalName](#Section_e98ee550643046808ac34bbc25a91a93) | Gets the name of the terminal associated with the specified session on the terminal server.  Opnum: 6 |
| [RpcGetState](#Section_58b06e1dbfe44a2f90a19c9129923e99) | Gets the state of the specified session on the terminal server.  Opnum: 7 |
| [RpcIsSessionDesktopLocked](#Section_0a12b501a52c41e682b306638cc0ae63) | Checks whether the specified session on the terminal server is locked.  Opnum: 8 |
| [RpcShowMessageBox](#Section_63915a11630d4ab686d11034fde1e386) | Displays a message box, with a specified message and title, in the target user session running on the terminal server.  Opnum: 9 |
| [RpcGetTimes](#Section_4a7d8496a6c44162adaf7897baa8c4ff) | Gets the connected, disconnected, and logged on time for the specified session on the terminal server.  Opnum: 10 |
| [RpcGetSessionCounters](#Section_eeda3f5626de4a6da0737ec5390d7dbd) | Returns the various performance counters associated with the terminal server.  Opnum: 11 |
| [RpcGetSessionInformation](#Section_1a7fa234480540d3a41c8b55bde2eda0) | Retrieves information about a specified session running on a terminal server.  Opnum: 12 |
| Opnum13NotUsedOnWire | Not implemented.  Opnum: 13 |
| Opnum14NotUsedOnWire | Not implemented.  Opnum: 14 |
| [RpcGetLoggedOnCount](#Section_8e02daa60b31494aa3c48c8d73faf442) | Gets the number of user-connected and device-connected sessions.  Opnum: 15 |
| [RpcGetSessionType](#Section_e7deb145e33f4ec0a2ac524b0381493d) | Gets the type of the specified session on a terminal server.  Opnum: 16 |
| [RpcGetSessionInformationEx](#Section_7a6b39aa61a34a89b486520e81f4f9e1) | Retrieves extended information about a specified session running on a terminal server.  Opnum: 17 |
| Opnum18NotUsedOnWire | Not implemented.  Opnum: 18 |
| Opnum19NotUsedOnWire | Not implemented.  Opnum: 19 |

##### RpcOpenSession (Opnum 0)

The RpcOpenSession method returns a [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to a specified session on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). No special permissions are required to call this method.

1. HRESULT RpcOpenSession(
2. [in] handle\_t hBinding,
3. [in] LONG SessionId,
4. [out] SESSION\_HANDLE\* phSession
5. );

**hBinding:**  The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle. For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**SessionId:**  The identifier of the session to open. This session MUST be present on the terminal server, or this call will fail. This MUST NOT be the session ID of any of the listener sessions.

**phSession:**  A handle to the session. This is of type [SESSION\_HANDLE](#Section_08a45954817343c6aa107aeda48e8dc2).

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcCloseSession (Opnum 1)

The RpcCloseSession method closes the connection to the specified session on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). This method MUST be called after [RpcOpenSession](#Section_56ede88b913b49119540389fddf879e5). The call to this method MUST be serialized if there are multiple threads running otherwise the behavior of this function is unknown. No special permissions are required to call this method.

1. HRESULT RpcCloseSession(
2. [in, out] SESSION\_HANDLE\* phSession
3. );

**phSession:** Pointer to a [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to the session to close. The pointer is returned by RpcOpenSession. This is of type [SESSION\_HANDLE](#Section_08a45954817343c6aa107aeda48e8dc2). The handle is set to NULL when the call returns.

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcConnect (Opnum 2)

The RpcConnect method reconnects a session [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) returned by [RpcOpenSession](#Section_56ede88b913b49119540389fddf879e5) to another specified session on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). This method MUST be called after RpcOpenSession. If the method succeeds, the state of the session is State\_Active as defined in the **WINSTATIONSTATECLASS** enumeration (section [2.2.1.9](#Section_b619d8aa4e8a4716954ea4d596ed37f8)).

The caller MUST have WINSTATION\_CONNECT permission to connect the current session and the caller MUST have WINSTATION\_DISCONNECT permission to disconnect the target session. For each aforementioned required permission, the method checks whether the caller has the permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting the Access Request mask to the specific permission, and fails if the caller does not have the permission.

1. HRESULT RpcConnect(
2. [in] SESSION\_HANDLE hSession,
3. [in] LONG TargetSessionId,
4. [in, string] WCHAR\* szPassword
5. );

**hSession:** The handle to a session returned by RpcOpenSession. This is of type [SESSION\_HANDLE](#Section_08a45954817343c6aa107aeda48e8dc2).

**TargetSessionId:** The identifier of the session on the terminal server to which to reconnect the session handle. This session MUST be present on the terminal server or this call will fail.

**szPassword:** The password of the user connected to the current session. This is an optional field. If not specified, the terminal server will impersonate the current user, making the call and checking whether it has permission to disconnect the current session.

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcDisconnect (Opnum 3)

The RpcDisconnect method disconnects the specified session on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). This method MUST be called after [RpcOpenSession](#Section_56ede88b913b49119540389fddf879e5). If the method succeeds, the state of the session is State\_Disconnected as defined in the **WINSTATIONSTATECLASS** enumeration (section [2.2.1.9](#Section_b619d8aa4e8a4716954ea4d596ed37f8)).

The caller MUST have WINSTATION\_DISCONNECT permission to disconnect the session. The method checks whether the caller has WINSTATION\_DISCONNECT permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. HRESULT RpcDisconnect(
2. [in] SESSION\_HANDLE hSession
3. );

**hSession:** The [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to the session returned by RpcOpenSession. This is of type [SESSION\_HANDLE](#Section_08a45954817343c6aa107aeda48e8dc2).

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcLogoff (Opnum 4)

The RpcLogoff method logs off the specified session on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). This method MUST be called after [RpcOpenSession](#Section_56ede88b913b49119540389fddf879e5). The caller MUST have WINSTATION\_LOGOFF permission to log off the session. The method checks whether the caller has WINSTATION\_LOGOFF permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. HRESULT RpcLogoff(
2. [in] SESSION\_HANDLE hSession
3. );

**hSession:** The [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to the session returned by RpcOpenSession. This is of type [SESSION\_HANDLE](#Section_08a45954817343c6aa107aeda48e8dc2).

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcGetUserName (Opnum 5)

The RpcGetUserName method gets the username and [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) name of the user logged on to the specified session on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). This method MUST be called after [RpcOpenSession](#Section_56ede88b913b49119540389fddf879e5). The caller MUST have WINSTATION\_QUERY permission for the session. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. HRESULT RpcGetUserName(
2. [in] SESSION\_HANDLE hSession,
3. [out, string] WCHAR\*\* pszUserName,
4. [out, string] WCHAR\*\* pszDomain
5. );

**hSession:** The [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to the session returned by RpcOpenSession. This is of type [SESSION\_HANDLE](#Section_08a45954817343c6aa107aeda48e8dc2).

**pszUserName:** The name of the user who is logged on to the specific session.

**pszDomain:** The domain to which the currently logged-on user belongs. If the terminal server is not joined to a domain, *pszDomain* will be the name of the terminal server computer.

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcGetTerminalName (Opnum 6)

The RpcGetTerminalName method gets the name of the terminal associated with the specified session on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). This method MUST be called after [RpcOpenSession](#Section_56ede88b913b49119540389fddf879e5). The caller MUST have WINSTATION\_QUERY permission for the session. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. HRESULT RpcGetTerminalName(
2. [in] SESSION\_HANDLE hSession,
3. [out, string] WCHAR\*\* pszTerminalName
4. );

**hSession:** The [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to the session returned by RpcOpenSession. This is of type [SESSION\_HANDLE](#Section_08a45954817343c6aa107aeda48e8dc2).

**pszTerminalName:** The name of the terminal associated with the specific session.

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcGetState (Opnum 7)

The RpcGetState method gets the state of the specified session on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). This method MUST be called after [RpcOpenSession](#Section_56ede88b913b49119540389fddf879e5). The caller MUST have WINSTATION\_QUERY permission for the session. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. HRESULT RpcGetState(
2. [in] SESSION\_HANDLE hSession,
3. [out] LONG\* plState
4. );

**hSession:** The [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to the session returned by RpcOpenSession. This is of type [SESSION\_HANDLE](#Section_08a45954817343c6aa107aeda48e8dc2).

**plState:**  The current state of the session as defined in [WINSTATIONSTATECLASS (section 2.2.1.9)](#Section_b619d8aa4e8a4716954ea4d596ed37f8).

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcIsSessionDesktopLocked (Opnum 8)

The RpcIsSessionDesktopLocked method checks whether the specified session on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) is in a locked state. This method MUST be called after [RpcOpenSession](#Section_56ede88b913b49119540389fddf879e5). The caller MUST have WINSTATION\_QUERY permission for the session. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. HRESULT RpcIsSessionDesktopLocked(
2. [in] SESSION\_HANDLE hSession
3. );

**hSession:** The [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to the session returned by RpcOpenSession. This is of type [SESSION\_HANDLE](#Section_08a45954817343c6aa107aeda48e8dc2).

**Return Values:** The method MUST return S\_OK (0x00000000) if the session is locked; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcShowMessageBox (Opnum 9)

The RpcShowMessageBox method displays a message box, with specified message and title, in the target user session running on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). This method MUST be called after [RpcOpenSession](#Section_56ede88b913b49119540389fddf879e5). The caller MUST have WINSTATION\_MSG permission for the session. The method checks whether the caller has WINSTATION\_MSG permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. HRESULT RpcShowMessageBox(
2. [in] SESSION\_HANDLE hSession,
3. [in, string] WCHAR\* szTitle,
4. [in, string] WCHAR\* szMessage,
5. [in] ULONG ulStyle,
6. [in] ULONG ulTimeout,
7. [out] ULONG\* pulResponse,
8. [in] BOOL bDoNotWait
9. );

**hSession:** The [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to the session returned by RpcOpenSession. This is of type [SESSION\_HANDLE](#Section_08a45954817343c6aa107aeda48e8dc2).

**szTitle:** The title to assign to the message box.

**szMessage:** The message to display inside the message box.

**ulStyle:** Specifies the contents and behavior of the message box. This parameter can be a combination of flags specified for the **uType** parameter of the MessageBox function as defined in [[MSDN-MSGBOX]](https://go.microsoft.com/fwlink/?LinkId=93359).

**ulTimeout:** The time in seconds for which to display the message box. This time-out value is managed by another system component which dismisses the message box if no user input is entered during this interval.

**pulResponse:** Pointer to a variable that receives the user's response, which can be one of the following values. The values defined in [MSDN-MSGBOX].

| Value | Meaning |
| --- | --- |
| IDABORT  3 | The Abort button was selected. |
| IDCANCEL  2 | The Cancel button was selected. |
| IDIGNORE  5 | The Ignore button was selected. |
| IDNO  7 | The No button was selected. |
| IDOK  1 | The OK button was selected. |
| IDRETRY  4 | The Retry button was selected. |
| IDYES  6 | The Yes button was selected. |
| IDASYNC  32001 | The *bDoNotWait* parameter was TRUE, so the function returned without waiting for a response. |
| IDTIMEOUT  32000 | The *bDoNotWait* parameter was FALSE and the time-out interval elapsed. |

**bDoNotWait:** Set to FALSE to wait for the message box to time-out or close, TRUE otherwise.

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcGetTimes (Opnum 10)

The RpcGetTimes method gets the connected, disconnected, and logged-on time for the specified session on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). This method MUST be called after [RpcOpenSession](#Section_56ede88b913b49119540389fddf879e5). The caller MUST have WINSTATION\_QUERY permission for the session. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. HRESULT RpcGetTimes(
2. [in] SESSION\_HANDLE hSession,
3. [out] hyper\* pConnectTime,
4. [out] hyper\* pDisconnectTime,
5. [out] hyper\* pLogonTime
6. );

**hSession:** Handle to the session returned by RpcOpenSession. This is of type [SESSION\_HANDLE](#Section_08a45954817343c6aa107aeda48e8dc2).

**pConnectTime:**  The most recent time of a connection to the session.

Time is measured as the number of 100-nanosecond intervals since January 1, 1601 (UTC).

**pDisconnectTime:** The most recent time of a disconnection from the session.

Time is measured as the number of 100-nanosecond intervals since January 1, 1601 (UTC).

**pLogonTime:** The most recent time of a logon to the session.

Time is measured as the number of 100-nanosecond intervals since January 1, 1601 (UTC).

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcGetSessionCounters (Opnum 11)

The **RpcGetSessionCounters** method returns the various performance counters associated with the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). No special permissions are required to call this method.

1. HRESULT RpcGetSessionCounters(
2. [in] handle\_t hBinding,
3. [in, out, size\_is(uEntries)] PTS\_COUNTER pCounter,
4. [in] ULONG uEntries
5. );

**hBinding:**  The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee). For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**pCounter:** An array of [TS\_COUNTER](#Section_a0173488c4834340ab1aaab2d21ea323) structures. The caller MUST set the **dwCounterId** field in the TS\_COUNTER structures for each entry in the array to indicate the counter whose current value to retrieve. On return, the method MUST set the value for that performance counter. If the performance counter ID is not recognized or is not supported, the method will set the **bResult** field to 0.

**uEntries:** The number of performance counters to query. Indicates the size of the array pointed to by the *pCounter* parameter.

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcGetSessionInformation (Opnum 12)

The RpcGetSessionInformation method retrieves information about a specified session running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). The caller MUST have WINSTATION\_QUERY permission for the session. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. HRESULT RpcGetSessionInformation(
2. [in] handle\_t hBinding,
3. [in] LONG SessionId,
4. [ref, out] PLSMSESSIONINFORMATION pSessionInfo
5. );

**hBinding:**  The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee). For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**SessionId:**  The identifier of the session whose information is to be retrieved.

**pSessionInfo:**  A [PLSMSESSIONINFORMATION](#Section_fadef65b0d6542b8ad14127106499f85) element containing information about the session.

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcGetLoggedOnCount (Opnum 15)

The RpcGetLoggedOnCount method gets the number of user-connected and device-connected sessions. No special permissions are required to call this method.

1. HRESULT RpcGetLoggedOnCount(
2. [in] handle\_t hBinding,
3. [out] ULONG\* pUserSessions,
4. [out] ULONG\* pDeviceSessions
5. );

**hBinding:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee). For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**pUserSessions:** The number of sessions that are of SESSIONTYPE\_REGULARDESKTOP, or SESSIONTYPE\_ALTERNATESHELL, or SESSIONTYPE\_REMOTEAPP as defined in [SESSIONTYPE (section 2.2.1.18)](#Section_12510151f01d4984a1bb64ec4298c31c).

**pDeviceSessions:** The number of sessions connected using media center extender device only. These sessions are of SESSIONTYPE\_MEDIACENTEREXT as defined in SESSIONTYPE (section 2.2.1.18). For more information on media center, see [[MSFT-WINMCE]](https://go.microsoft.com/fwlink/?LinkId=208408).

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcGetSessionType (Opnum 16)

The RpcGetSessionType method gets the type associated with the specified session. No special permissions are required to call this method.[<149>](#Appendix_A_149" \o "Product behavior note 149)

1. HRESULT RpcGetSessionType(
2. [in] handle\_t hBinding,
3. [in] LONG\* SessionId,
4. [out] ULONG\* pSessionType
5. );

**hBinding:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee). For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**SessionId:** The identifier of the session whose type is being retrieved.

**pSessionType:** The type of the session as defined in [SESSIONTYPE (section 2.2.1.18)](#Section_12510151f01d4984a1bb64ec4298c31c).

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcGetSessionInformationEx (Opnum 17)

The RpcGetSessionInformationEx method retrieves extended information about a specified session running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998).[<150>](#Appendix_A_150" \o "Product behavior note 150) The caller MUST have WINSTATION\_QUERY permission for the session. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. HRESULT RpcGetSessionInformationEx(
2. [in] handle\_t hBinding,
3. [in] LONG SessionId,
4. [in] DWORD Level,
5. [ref, out] PLSMSESSIONINFORMATION\_EX LSMSessionInfoExPtr
6. );

**hBinding:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee). For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**Level:** The level of the information to be retrieved. This MUST be 1.

**LSMSessionInfoExPtr:** A [PLSMSESSIONINFORMATION\_EX](#Section_4e7039f4d1d044f282f008b057476db5) element containing information about the session.

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

#### TermSrvNotification

The TermSrvNotification, or LSM Notification interface, provides methods that manage asynchronous operations. Methods that initiate asynchronous operations return a pointer to an LSM Notification (TermSrvNotification) interface, allowing the caller to optionally cancel, or wait for, the status of the asynchronous operation. The version for this interface is 1.0.

For information about [**endpoints**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee), [**UUID**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) values, and versions, see sections [2.1](#Section_ac0535d0a8474386ae3ee93a50ff2484) and [1.9](#Section_4f5ff56828874c508b3638bb85170b58).

Methods in RPC Opnum Order

| Method | Description |
| --- | --- |
| [RpcWaitForSessionState](#Section_637ac10c6570460380223c34ea2ea73a) | Blocks until the state of the specified session running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) changes to the desired state.  Opnum: 0 |
| [RpcRegisterAsyncNotification](#Section_bb3d1c58f397472e82ed44c242846464) | Registers for an event or events happening on a terminal server.  Opnum: 1 |
| [RpcWaitAsyncNotification](#Section_81d5ba342ae446149aca8deb1eb7a0a1) | Starts the wait for the specified notification to be signaled by the terminal server.  Opnum: 2 |
| [RpcUnRegisterAsyncNotification](#Section_886923e61d674ad2a79ac24c9398b224) | Cancels the asynchronous operation of waiting for notification from the terminal server.  Opnum: 3 |

##### RpcWaitForSessionState (Opnum 0)

The RpcWaitForSessionState method blocks until the state of the specified session running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) changes to the desired state. The caller MUST have WINSTATION\_QUERY permission for the session. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. HRESULT RpcWaitForSessionState(
2. [in] handle\_t hBinding,
3. [in] LONG SessionId,
4. [in] LONG State,
5. [in] ULONG Timeout
6. );

**hBinding:**  The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee). For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**SessionId:**  The ID of the session for which to await state change. This MUST NOT be the session ID of any of the [**listener**](#gt_1e5aa0c2-4617-4d7a-969e-a7091c415c8d) sessions.

**State:**  The desired state of the session, as specified in [WINSTATIONSTATECLASS (section 2.2.1.9)](#Section_b619d8aa4e8a4716954ea4d596ed37f8) with the exception of State\_Idle and State\_Listen, for which to wait. The call will return when the session changes to this state.

**Timeout:**  Maximum time, in milliseconds, to wait for the call to return.

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcRegisterAsyncNotification (Opnum 1)

The RpcRegisterAsyncNotification method registers for an event or events happening on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). The caller MUST call [RpcWaitAsyncNotification](#Section_81d5ba342ae446149aca8deb1eb7a0a1) after calling RpcRegisterAsyncNotification to wait for the notification. No special permissions are required to call this method.

1. HRESULT RpcRegisterAsyncNotification(
2. [in] handle\_t hBinding,
3. [in] LONG SessionId,
4. [in] TNotificationId Mask,
5. [out] NOTIFY\_HANDLE\* phNotify
6. );

**hBinding:**  The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee). For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**SessionId:**  The ID of the session for which to wait for notification. If the value is -1, the call will register for notification for all sessions running on the terminal server. This MUST NOT be the session ID of any of the [**listener**](#gt_1e5aa0c2-4617-4d7a-969e-a7091c415c8d) sessions.

**Mask:**  Specifies the type of notification to wait for. This is of the type [TNotificationId](#Section_f8f4a0d198da4125994d206d4198f03c).

**phNotify:** Handle to the notification object. For more information, see [NOTIFY\_HANDLE](#Section_557683a82f4a416abb15fa353fdc4aed).

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcWaitAsyncNotification (Opnum 2)

The RpcWaitAsyncNotification method starts the wait for the specified [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) notification. The notification object specified in [RpcRegisterAsyncNotification](#Section_bb3d1c58f397472e82ed44c242846464) is called by RPC when a notification occurs. This is asynchronous notification and RpcWaitAsyncNotification starts the wait for notification and returns. Specify the notification object using RpcRegisterAsyncNotification and then start the notification wait process using RpcWaitAsyncNotification. No special permissions are required to call this method.

1. HRESULT RpcWaitAsyncNotification(
2. [in] NOTIFY\_HANDLE hNotify,
3. [out, size\_is(, \*pEntries)] PSESSION\_CHANGE\* SessionChange,
4. [out] ULONG\* pEntries
5. );

**hNotify:**  The notification [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) returned by RpcRegisterAsyncNotification.

**SessionChange:**  An array of type [SESSION\_CHANGE](#Section_e22785c95a374a47864aed2865d053bc) containing information about all the sessions and the notifications received for that session.

**pEntries:**  The number of entries returned in the *SessionChange* array.

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcUnRegisterAsyncNotification (Opnum 3)

The RpcUnRegisterAsyncNotification method cancels the asynchronous operation of waiting for notification from the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). This MUST be called after [RpcRegisterAsyncNotification](#Section_bb3d1c58f397472e82ed44c242846464). The call to this method MUST be serialized if there are multiple threads running otherwise the behavior of this function is unknown. No special permissions are required to call this method.

1. HRESULT RpcUnRegisterAsyncNotification(
2. [in, out] NOTIFY\_HANDLE\* phNotify
3. );

**phNotify:** The notification [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) returned by RpcRegisterAsyncNotification. This is of type [NOTIFY\_HANDLE](#Section_557683a82f4a416abb15fa353fdc4aed).

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

#### TermSrvEnumeration

The TermSrvEnumeration provides methods for enumerating sessions and session information. The version for this interface is 1.0.

For information about endpoints, [**UUID**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) values, and versions, see sections [2.1](#Section_ac0535d0a8474386ae3ee93a50ff2484) and [1.9](#Section_4f5ff56828874c508b3638bb85170b58).

Methods in RPC Opnum Order

| Method | Description |
| --- | --- |
| [RpcOpenEnum](#Section_f17f4b8d8d524538acf3cdfbe581a99b) | Returns a [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) of type [ENUM\_HANDLE](#Section_15f02d59200a435da07b055e1276d01c), which can be used to query information about the sessions currently running on a terminal server.  Opnum: 0 |
| [RpcCloseEnum](#Section_cb9386d5d99e46ff8199563f6b59ecb8) | Closes the enumeration object returned by RpcOpenEnum. This method MUST be called after RpcOpenEnum.  Opnum: 1 |
| [RpcFilterByState](#Section_14ed1d6871f2449599460116f87ee1ee) | Based on the state of the sessions, adds a filter to the session enumeration result running on a terminal server.  Opnum: 2 |
| [RpcFilterByCallersName](#Section_7672f2e67b124dc290b0ad43a231d83e) | Based on the caller name, adds a filter to the session enumeration result running on a terminal server.  Opnum: 3 |
| [RpcEnumAddFilter](#Section_2cd006cdfb1043a6a16db01379f9cb53) | Adds another filter to the current enumeration.  Opnum: 4 |
| [RpcGetEnumResult](#Section_1a7d5d1d1ce5448fbb4ac79741982edb) | Returns a structure of the type [PSESSIONENUM](#Section_fd8759ef4f6c45c6b5429d5e53e90f7d) containing the list of sessions currently running on the terminal server after applying the specified filter.  Opnum: 5 |
| [RpcFilterBySessionType](#Section_fd4db5fa13b3490b95283cb108544a4e) | Based on the type of the session, adds a filter to the session enumeration result running on a terminal server.  Opnum: 6 |
| Opnum7NotUsedOnWire | Not implemented.  Opnum: 7 |
| [RpcGetSessionIds](#Section_922264ed0a2d441fa71c2d3ffa975814) | Returns a list of the IDs associated with the sessions running on a terminal server that satisfies the specified filter.  Opnum: 8 |
| [RpcGetEnumResultEx](#Section_cb2be08441dd44ec81dc52d371a28554) | Returns a structure of [PSESSIONENUM\_EX](#Section_35a722eb00a04ba08e4c735d06378675) containing the list of sessions currently running on the terminal server after applying the specified filter.  Opnum: 9 |
| [RpcGetAllSessions](#Section_68e824e0d78649878ccbc34503fffd3f) | Returns a structure of [PEXECENVDATA](#Section_20bf286ea1584d5fa9f495eeaf1a4869) containing the list of sessions currently running on the terminal server and the sessions currently running on the virtual machines hosted by the server.  Opnum: 10 |
| [RpcGetAllSessionsEx](#Section_3f72497925cd41d89829244f4425f9c6) | Returns a structure of [PEXECENVDATAEX](#Section_0a3f7e7b03f84f62a65fc2d670bb3173) containing the list of sessions currently running on the terminal server and the sessions currently running on the virtual machines hosted by the server.  Opnum: 11 |

##### RpcOpenEnum (Opnum 0)

The RpcOpenEnum method returns a [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) of the type [ENUM\_HANDLE](#Section_15f02d59200a435da07b055e1276d01c), which can be used to query information about the sessions that are currently running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). No special permissions are required to call this method. However, only sessions to which the caller has WINSTATION\_QUERY permission are enumerated.

1. HRESULT RpcOpenEnum(
2. [in] handle\_t hBinding,
3. [out] ENUM\_HANDLE\* phEnum
4. );

**hBinding:**  The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle. For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**phEnum:**  The handle to the session enumeration object. This is of type ENUM\_HANDLE.

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcCloseEnum (Opnum 1)

The RpcCloseEnum method closes the enumeration object returned by [RpcOpenEnum](#Section_f17f4b8d8d524538acf3cdfbe581a99b). This method MUST be called after RpcOpenEnum. No special permissions are required to call this method.

1. HRESULT RpcCloseEnum(
2. [in, out] ENUM\_HANDLE\* phEnum
3. );

**phEnum:**  The [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to the session enumeration object. This is of type [ENUM\_HANDLE](#Section_15f02d59200a435da07b055e1276d01c).

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcFilterByState (Opnum 2)

The RpcFilterByState method adds a filter to the session enumeration result, running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998), based on the state of the sessions. This method MUST be called after [RpcOpenEnum](#Section_f17f4b8d8d524538acf3cdfbe581a99b) and before [RpcGetEnumResult](#Section_1a7d5d1d1ce5448fbb4ac79741982edb) or [RpcGetEnumResultEx](#Section_cb2be08441dd44ec81dc52d371a28554). No special permissions are required to call this method.

1. HRESULT RpcFilterByState(
2. [in] ENUM\_HANDLE hEnum,
3. [in] LONG State,
4. [in] BOOL bInvert
5. );

**hEnum:**  The [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to the session enumeration object. This is of type [ENUM\_HANDLE](#Section_15f02d59200a435da07b055e1276d01c).

**State:** The session state, as specified in section [3.3.4.1.8](#Section_58b06e1dbfe44a2f90a19c9129923e99), to be used to filter out the enumeration result. Only the sessions with the specified state will be returned.

**bInvert:**  Set to TRUE to imply that the result of the comparison during enumeration will be inverted, FALSE otherwise.

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcFilterByCallersName (Opnum 3)

The RpcFilterByCallersName method adds a filter to the session enumeration result, running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998), based on the caller name. The enumeration will return only sessions belonging to the user making this call. This method MUST be called after [RpcOpenEnum](#Section_f17f4b8d8d524538acf3cdfbe581a99b) and before [RpcGetEnumResult](#Section_1a7d5d1d1ce5448fbb4ac79741982edb) or [RpcGetEnumResultEx](#Section_cb2be08441dd44ec81dc52d371a28554). No special permissions are required to call this method.

1. HRESULT RpcFilterByCallersName(
2. [in] ENUM\_HANDLE hEnum
3. );

**hEnum:**  The [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to the session enumeration object. This is of type [ENUM\_HANDLE](#Section_15f02d59200a435da07b055e1276d01c).

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcEnumAddFilter (Opnum 4)

The RpcEnumAddFilter method adds another filter to the current enumeration. This method MUST be called after [RpcOpenEnum](#Section_f17f4b8d8d524538acf3cdfbe581a99b) and before [RpcGetEnumResult](#Section_1a7d5d1d1ce5448fbb4ac79741982edb) or [RpcGetEnumResultEx](#Section_cb2be08441dd44ec81dc52d371a28554). No special permissions are required to call this method.

1. HRESULT RpcEnumAddFilter(
2. [in] ENUM\_HANDLE hEnum,
3. [in] ENUM\_HANDLE hSubEnum
4. );

**hEnum:**  The [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to the session enumeration object. This is of type [ENUM\_HANDLE](#Section_15f02d59200a435da07b055e1276d01c).

**hSubEnum:**  The handle to another enumeration whose filter will be applied to the current enumeration. The other enumeration MUST be created using RpcOpenEnum and any combination of [RpcFilterByState](#Section_14ed1d6871f2449599460116f87ee1ee), [RpcFilterByCallersName](#Section_7672f2e67b124dc290b0ad43a231d83e), and [RpcFilterBySessionType](#Section_fd4db5fa13b3490b95283cb108544a4e).

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcGetEnumResult (Opnum 5)

The RpcGetEnumResult method returns a pointer of the type [PSESSIONENUM](#Section_fd8759ef4f6c45c6b5429d5e53e90f7d) which points to the list of sessions currently running on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) after applying the specified filter. This method MUST be called after [RpcOpenEnum](#Section_f17f4b8d8d524538acf3cdfbe581a99b). No special permissions are required to call this method. However, only sessions for which the caller has WINSTATION\_QUERY permission are enumerated. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and skips the sessions for which the caller does not have the permission.

1. HRESULT RpcGetEnumResult(
2. [in] ENUM\_HANDLE hEnum,
3. [out, size\_is(,\*pEntries)] PSESSIONENUM\* ppSessionEnumResult,
4. [in] DWORD Level,
5. [out] ULONG\* pEntries
6. );

**hEnum:**  The [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to the session enumeration object. This is of type [ENUM\_HANDLE](#Section_15f02d59200a435da07b055e1276d01c).

**ppSessionEnumResult:** A pointer of the type PSESSIONENUM which points to the list of sessions currently running on the terminal server.

**Level:** The level of information requested. This field MUST be set to the supported values of 1 or 2. If the server does not support the level requested, it will set the level to the maximum supported level and return information accordingly.[<151>](#Appendix_A_151" \o "Product behavior note 151)

**pEntries:**  The number of sessions currently running on the terminal server and returned in the *ppSessionEnumResult* structure.

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcFilterBySessionType (Opnum 6)

The RpcFilterBySessionType method adds a filter to the session enumeration result, running on a terminal server, based on the type of the session. This method MUST be called after [RpcOpenEnum](#Section_f17f4b8d8d524538acf3cdfbe581a99b) and before [RpcGetEnumResult](#Section_1a7d5d1d1ce5448fbb4ac79741982edb) or [RpcGetEnumResultEx](#Section_cb2be08441dd44ec81dc52d371a28554). No special permissions are required to call this method.

1. HRESULT RpcFilterBySessionType(
2. [in] ENUM\_HANDLE hEnum,
3. [in] GUID\* pSessionType
4. );

**hEnum:**  The [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to the session enumeration object. This is of type [ENUM\_HANDLE](#Section_15f02d59200a435da07b055e1276d01c).

**pSessionType:** The session GUID to be used to filter out the enumeration result. Only the session with the specified GUID will be returned.

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcGetSessionIds (Opnum 8)

The RpcGetSessionIds method returns a list of the IDs associated with the sessions running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) that satisfy the specified filter. No special permissions are required to call this method. However, only sessions for which the caller has WINSTATION\_QUERY permission are enumerated. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and skips sessions for which the caller does not have the permission.

1. HRESULT RpcGetSessionIds(
2. [in] handle\_t hBinding,
3. [in] SESSION\_FILTER Filter,
4. [in, range(0, 0xFFFF)] ULONG MaxEntries,
5. [out, size\_is(,\*pcSessionIds)] LONG\*\* pSessionIds,
6. [out] ULONG\* pcSessionIds
7. );

**hBinding:**  The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee). For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**Filter:**  The filter to apply to the sessions running on the terminal server. This is of type [SESSION\_FILTER](#Section_352695847c5e441081951aa49559274e).

**MaxEntries:**  The maximum number of session IDs to return. This value MUST be less than or equal to the number of entries in the *pSessionIds* array.

**pSessionIds:**  An array to contain the list of session IDs running on the terminal server and filtered as specified by *Filter*.

**pcSessionIds:**  The number of session IDs returned.

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcGetEnumResultEx (Opnum 9)

The RpcGetEnumResultEx method returns a pointer of the type [PSESSIONENUM\_EX](#Section_35a722eb00a04ba08e4c735d06378675), which points to the list of sessions currently running on the terminal server after applying the specified filter. This method MUST be called after [RpcOpenEnum](#Section_f17f4b8d8d524538acf3cdfbe581a99b). No special permissions are required to call this method. However, only sessions for which the caller has WINSTATION\_QUERY permission are enumerated. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and skips the sessions for which the caller does not have the permission.

1. HRESULT RpcGetEnumResultEx(
2. [in] ENUM\_HANDLE hEnum,
3. [out, size\_is(,\*pEntries)] PSESSIONENUM\_EX\* ppSessionEnumResult,
4. [in] DWORD Level,
5. [out] ULONG\* pEntries
6. );

**hEnum:** The [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to the session enumeration object. This is of type [ENUM\_HANDLE](#Section_15f02d59200a435da07b055e1276d01c).

**ppSessionEnumResult:** A pointer of the type PSESSIONENUM\_EX which points to the list of sessions currently running on the terminal server.

**Level:** The level of information requested. This field MUST be set to the supported values of 1, 2, or 3. If the server does not support the level requested, it will set the level to the maximum supported level and return information accordingly.[<152>](#Appendix_A_152" \o "Product behavior note 152)

| Value | Meaning |
| --- | --- |
| 1 | The union [SessionInfo\_Ex](#Section_1cae32cd3bd3436db9501f43c2b40863) will have the [SessionEnum\_Level1](#Section_0c4fde3b4fdb45248cae792d0320247a) structure. |
| 2 | The union SessionInfo\_Ex will have the [SessionEnum\_Level2](#Section_1b889bfe8d324cbabd92f9d7c09d5f60) structure. |
| 3 | The union SessionInfo\_Ex will have the [SessionEnum\_Level3](#Section_d1662f979e3a4fbbbdd52050df69e902) structure. |

**pEntries:** The number of sessions currently running on the terminal server and returned in the **ppSessionEnumResult** structure.

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcGetAllSessions (Opnum 10)

The RpcGetAllSessions method returns a pointer of type [PEXECENVDATA](#Section_20bf286ea1584d5fa9f495eeaf1a4869), which points to the list of sessions currently running on the terminal server and the sessions running on various virtual machines hosted by the server. No special permissions are required to call this method. However, only sessions for which the caller has WINSTATION\_QUERY permission are enumerated. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and skips the sessions for which the caller does not have the permission.[<153>](#Appendix_A_153" \o "Product behavior note 153)

1. HRESULT RpcGetAllSessions(
2. [in] handle\_t hBinding,
3. [in, out] ULONG\* pLevel,
4. [out, size\_is(,\*pcEntries)] PEXECENVDATA\* ppSessionData,
5. [out] ULONG\* pcEntries
6. );

**hBinding:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee). For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**pLevel:** The level of information requested. This field MUST be set to the supported values of 1 or 2. If the server does not support the level requested, it will set the level to the maximum supported level and return an implementation-specific nonzero value.

| Value | Meaning |
| --- | --- |
| 1 | The union [ExecEnvData](#Section_9846fe50582c4f69a070677049ebc1ae) will have the [EXECENVDATA\_LEVEL1](#Section_004305a642654ff5bb8a58fd0f7cf2d5) structure. |
| 2 | The union ExecEnvData will have the [EXECENVDATA\_LEVEL2](#Section_e85daed7ed774c8aafc05596c8d41c9c) structure. |

**ppSessionData:** A pointer of type PEXECENVDATA (section 2.2.2.6), which points to the list of sessions currently running on the terminal server and the session running on virtual machines hosted by the server.[<154>](#Appendix_A_154" \o "Product behavior note 154)

**pcEntries:** The number of sessions currently running on the terminal server and returned in the **ppSessionData** structure.

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcGetAllSessionsEx (Opnum 11)

The **RpcGetAllSessionsEx** method returns a pointer of type [PEXECENVDATAEX](#Section_0a3f7e7b03f84f62a65fc2d670bb3173), which points to the list of sessions currently running on the terminal server and the sessions running on various virtual machines hosted by the server. No special permissions are required to call this method. However, only sessions for which the caller has WINSTATION\_QUERY permission are enumerated. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and skips the sessions for which the caller does not have the permission.[<155>](#Appendix_A_155" \o "Product behavior note 155)

1. HRESULT RpcGetAllSessionsEx(
2. [in] handle\_t hBinding,
3. [in] ULONG Level,
4. [out, size\_is(,\*pcEntries)] PEXECENVDATAEX\* ppSessionData,
5. [out] ULONG\* pcEntries
6. );

**hBinding:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee). For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**Level:** The level of information requested. This field MUST be set to 1. If the server does not support the level requested, it will set the level to the maximum supported level and return an implementation-specific nonzero value.

| Value | Meaning |
| --- | --- |
| 1 | The union [ExecEnvDataEx](#Section_d77326bf8ed74e1d9429965c2cd97f37) has the [EXECENVDATAEX\_LEVEL1](#Section_b4295b0ec64140abbbdbea44323b0804) structure. |

**ppSessionData:** A pointer of type PEXECENVDATAEX (section 2.2.2.7), which points to the list of sessions currently running on the terminal server and the sessions running on virtual machines hosted by the server.

**pcEntries:** The number of sessions currently running on the terminal server and returned in the **ppSessionData** structure.

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

### Timer Events

No protocol timer events are required on the client except the timers that are required in the underlying [**RPC transport**](#gt_c2eeb200-3cd0-4916-966e-d7d6bff1737a).

### Other Local Events

No local events are used on the server except the events maintained in the underlying [**RPC transport**](#gt_c2eeb200-3cd0-4916-966e-d7d6bff1737a).

## TermSrv Client Details

### Abstract Data Model

None.

### Timers

No protocol timers are required except those used internally by [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) to implement resiliency to network outages, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 3.3.3.2.1.

### Initialization

The client MUST create an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) connection to the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) using the details specified in section [2.1](#Section_ac0535d0a8474386ae3ee93a50ff2484).

### Processing Events and Sequencing Rules

When a method completes, the values returned by [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) MUST be returned unmodified to the upper layer.

### Timer Events

None.

### Other Local Events

None.

## TermSrv Server Details

### Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

#### Abstract Data Types

**PROTOCOLSTATUS\_INFO\_TYPE**: Specifies the protocol status information requested for a particular session running on a terminal server. It is defined in section [2.2.2.2](#Section_dd616139394c4ac281c7175024327604). The following list shows how **PROTOCOLSTATUS\_INFO\_TYPE** is used with various methods.

* [RpcGetProtocolStatus](#Section_954007d7e5ec483a82afd66ac8c647df): as the *InfoType* parameter.
* [RpcGetSessionProtocolLastInputTime](#Section_03f9b3dc041447f496da5a49e6c1e635): as the *InfoType* parameter.

**QUERY\_SESSION\_DATA\_TYPE**: The type of data to retrieve about the session. It is used with [RpcQuerySessionData](#Section_8661c3aabc4741b996625762ee052c2f) and MUST be set to one of the values described in section [2.2.2.3](#Section_8244f1b1ce6246f68b62949abd789b0f).

**WINSTATIONCLIENT**: Structure that defines the client-requested configuration when connecting to a session. This structure is described in detail in section [2.2.2.19](#Section_673d8ac0f55748cb98a649925160d729). It is used with the [RpcGetClientData](#Section_7f7d695c97f84b219ebf4580a8dab066) method.

**PLISTENERENUM**: Structure that contains information about a terminal server listener. This structure is described in detail in section [2.2.2.12](#Section_3c46804ffff447799f961ceb9a6c9559). It is used with [RpcGetAllListeners](#Section_4655b71d39b747258e78175cb5608195) and returned as the *ppListeners* parameter.

**WINSTATIONCONFIG**: Structure that contains WinStation configuration data. This structure is described in detail in section [2.2.2.30.1](#Section_379fe706cf87417eb1f2e8536013132d). It is used with the [RpcGetConfigData](#Section_6c4dc21eea534785b50593d23604f2c8) method.

**PROTOCOLSTATUS**: The status of the protocol used by the session. It is defined in section [2.2.2.20.1](#Section_c9066753acbd46789a728fb1b080bd09). It is used with the RpcGetProtocolStatus and RpcGetSessionProtocolLastInputTime methods.

**PROTOCOLSTATUSEX**: Structure defines the extended status of the protocol used by the session. This structure is described in detail in section [2.2.2.20.1.1](#Section_e32b64ed22e245cab9549e1b52a16c78). It is used with the RpcGetProtocolStatus and RpcGetSessionProtocolLastInputTime methods.

**WDCONFIG**: Structure containing the WinStation (session) driver configuration. This structure is described in detail in section [2.2.2.27](#Section_d1bf099beb544ed9a7230b1062dbc128). It is used with the RpcQuerySessionData method.

**WINSTATIONCONFIG2**: Structure that contains WinStation configuration data. This structure is described in detail in section [2.2.2.30](#Section_38d998ad51e744358231fff55348fff8). It is used with the RpcGetConfigData method.

**CACHE\_STATISTICS**: Structure that represents cache statistics on the protocol. This structure is described in detail in section [2.2.2.20.1.3](#Section_81203ca2e58b4681affa924e59671b5c). It is used with the RpcGetProtocolStatus method.

**PROTOCOLCOUNTERS**: Structure that represents the status of the protocol used by the system. This structure is described in detail in section [2.2.2.20.1.2](#Section_c5d7f335bfd9444d87a5b0d028a52b5e). It is used with the RpcGetProtocolStatus method.

**WINSTATIONVALIDATIONINFORMATION**: Structure that contains information that is required to identify and validate a per-device [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) (TS) [**client access license (CAL)**](#gt_9b06b9f2-1680-41c7-b89c-e08f9fe92a25) associated with the session. This structure is described in detail in section [2.2.2.51](#Section_5c439d8db58b4b0da2e5467aa05f993b). It is used with the RpcQuerySessionData method.

**RCM\_REMOTEADDRESS**: Structure that defines a remote address. This structure is described in detail in section [2.2.2.43](#Section_3c5dea7cf9a441c6b9f831b5bc985b6b). It is used with the [RpcGetRemoteAddress](#Section_4ee5f612b74c4e47811ce57edaf02c5b) method.

### Timers

**MaxConnectionTime**: The maximum allowed connection time setting of the session, in milliseconds. The session will disconnect/log off when the limit is reached.

**MaxDisconnectionTime**: The maximum allowed disconnect time of the session, in milliseconds. The session will disconnect/log off when the limit is reached.

**MaxIdleTime**: The maximum allowed idle time setting of the session, in milliseconds. The session will disconnect/log off when the limit is reached.

### Initialization

The parameters necessary to initialize the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) protocol are specified in section [2.1](#Section_ac0535d0a8474386ae3ee93a50ff2484).

### Message Processing Events and Sequencing Rules

This protocol asks the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) runtime to perform a strict NDR data consistency check at target level 6.0 for all methods unless otherwise specified, as defined in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 1.3.

When a method completes, the values returned by RPC MUST be returned unmodified to the upper layer. The methods MAY throw an exception and the [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) client MUST handle these exceptions by returning the unmodified exception code to the upper layer.

#### RCMPublic

The RCMPublic interface methods provide data about clients and sessions, and enable shadowing sessions. The version for this interface is 1.0.

For information about [**endpoints**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee), [**UUID**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) values, and versions, see sections [2.1](#Section_ac0535d0a8474386ae3ee93a50ff2484) and [1.9](#Section_4f5ff56828874c508b3638bb85170b58).

Methods in RPC Opnum Order

| Method | Description |
| --- | --- |
| [RpcGetClientData](#Section_7f7d695c97f84b219ebf4580a8dab066) | Returns information about the client that is connected to a particular session running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998).  Opnum: 0 |
| [RpcGetConfigData](#Section_6c4dc21eea534785b50593d23604f2c8) | Returns the configuration data that is associated with the user connected to a particular session running on a terminal server.  Opnum: 1 |
| [RpcGetProtocolStatus](#Section_954007d7e5ec483a82afd66ac8c647df) | Retrieves information about the status of the protocol that is used to connect to a particular session running on a terminal server.  Opnum: 2 |
| [RpcGetLastInputTime](#Section_fd64f3b50b1e4ed6a1fa81d7fea553fd) | Returns the time the last user input was received for the specified session running on a terminal server by the associated protocol.  Opnum: 3 |
| [RpcGetRemoteAddress](#Section_4ee5f612b74c4e47811ce57edaf02c5b) | Retrieves the IP address of the client machine that is connected to the session on the terminal server.  Opnum: 4 |
| Opnum5NotUsedOnWire | Not implemented.  Opnum: 5 |
| Opnum6NotUsedOnWire | Not implemented.  Opnum: 6 |
| Opnum7NotUsedOnWire | Not implemented.  Opnum: 7 |
| [RpcGetAllListeners](#Section_4655b71d39b747258e78175cb5608195) | Returns a list of all Terminal Services listeners running on a terminal server.  Opnum: 8 |
| [RpcGetSessionProtocolLastInputTime](#Section_03f9b3dc041447f496da5a49e6c1e635) | Returns the protocol status and time the last input was received by the protocol associated with a specific session running on a terminal server.  Opnum: 9 |
| [RpcGetUserCertificates](#Section_aa5f03ee828448f0a3e03c4661336541) | Returns the client X509 certificate used to connect to a session running on a terminal server.  Opnum: 10 |
| [RpcQuerySessionData](#Section_8661c3aabc4741b996625762ee052c2f) | Returns information about a particular session running on a terminal server.  Opnum: 11 |

##### RpcGetClientData (Opnum 0)

The RpcGetClientData method returns information about the client that is connected to a particular session running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). The caller must have WINSTATION\_QUERY permission. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. HRESULT RpcGetClientData(
2. [in] handle\_t hBinding,
3. [in] ULONG SessionId,
4. [out, size\_is(,\*pOutBuffByteLen)]
5. unsigned char\*\* ppBuff,
6. [out] ULONG\* pOutBuffByteLen
7. );

**hBinding:**  The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee). For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**SessionId:**  The ID of the session from which client data is to be retrieved.

**ppBuff:**  The buffer that contains the client data. This data is of type [PWINSTATIONCLIENT](#Section_673d8ac0f55748cb98a649925160d729), specified in section 2.2.2.19. The buffer is allocated by RpcGetClientData.

**pOutBuffByteLen:**  The number of bytes of client data that is returned.

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcGetConfigData (Opnum 1)

The RpcGetConfigData method returns the configuration data associated with the user connected to a particular session running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). The caller MUST have WINSTATION\_QUERY permission. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.[<156>](#Appendix_A_156" \o "Product behavior note 156)

1. HRESULT RpcGetConfigData(
2. [in] handle\_t hBinding,
3. [in] ULONG SessionId,
4. [out, size\_is(,\*pOutBuffByteLen )]
5. unsigned char\*\* ppBuff,
6. [out] ULONG\* pOutBuffByteLen
7. );

**hBinding:**  The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee). For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**SessionId:**  The ID of the session for which the client configuration data is to be retrieved.

**ppBuff:**  The buffer that will contain the client configuration data. This will be allocated by RpcGetConfigData. This data is of type [PWINSTATIONCONFIG](#Section_379fe706cf87417eb1f2e8536013132d).

**pOutBuffByteLen:**  The number of bytes of client configuration data that is returned.

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcGetProtocolStatus (Opnum 2)

The RpcGetProtocolStatus method retrieves information about the status of the protocol used to connect to a particular session running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). The caller MUST have WINSTATION\_QUERY permission for the session. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. HRESULT RpcGetProtocolStatus(
2. [in] handle\_t hBinding,
3. [in] ULONG SessionId,
4. [in] PROTOCOLSTATUS\_INFO\_TYPE InfoType,
5. [out, size\_is(,\*pcbProtoStatus )]
6. unsigned char\*\* ppProtoStatus,
7. [out] ULONG\* pcbProtoStatus
8. );

**hBinding:**  The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee). For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**SessionId:**  The ID of the session for which protocol status is to be retrieved.

**InfoType:**  Specifies what type of information to gather. This is of the type [PROTOCOLSTATUS\_INFO\_TYPE](#Section_dd616139394c4ac281c7175024327604).

**ppProtoStatus:**  The buffer that will contain protocol status data. This data is of the type [PROTOCOLSTATUS](#Section_c9066753acbd46789a728fb1b080bd09).

**pcbProtoStatus:**  The number of bytes of protocol data that is returned.

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion |

##### RpcGetLastInputTime (Opnum 3)

The RpcGetLastInputTime method returns the time the last user input was received by the associated protocol for the specified sessions running on a terminal server. The caller MUST have WINSTATION\_QUERY permission for the session. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. HRESULT RpcGetLastInputTime(
2. [in] handle\_t hBinding,
3. [in] ULONG SessionId,
4. [out] hyper\* pLastInputTime
5. );

**hBinding:**  The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee). For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**SessionId:**  The ID of the session for which the last user input time is to be retrieved.

**pLastInputTime:**  The time when the last user input was received by the server. This is a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601 (UTC).

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion |

##### RpcGetRemoteAddress (Opnum 4)

The RpcGetRemoteAddress method retrieves the IP address of the client computer connected to the session on the terminal server. The caller MUST have WINSTATION\_QUERY permission for the session. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. HRESULT RpcGetRemoteAddress(
2. [in] handle\_t hBinding,
3. [in] ULONG SessionId,
4. [out] PRCM\_REMOTEADDRESS pRemoteAddress
5. );

**hBinding:**  The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee). For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**SessionId:**  The ID of the session for which client data is to be retrieved.

**pRemoteAddress:** The address of the client computer that is connected to the session. This is of the type [PRCM\_REMOTEADDRESS](#Section_3c5dea7cf9a441c6b9f831b5bc985b6b).

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcGetAllListeners (Opnum 8)

The RpcGetAllListeners method returns a list of all [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) [**listeners**](#gt_1e5aa0c2-4617-4d7a-969e-a7091c415c8d) running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). No special permissions are required to call this method. However, only listeners for which the caller has WINSTATION\_QUERY permission are enumerated.

1. HRESULT RpcGetAllListeners(
2. [in] handle\_t hBinding,
3. [out, size\_is(,\*pNumListeners)]
4. PLISTENERENUM\* ppListeners,
5. [in] DWORD Level,
6. [out] ULONG\* pNumListeners
7. );

**hBinding:**  The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee). For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**ppListeners:**  The list of Terminal Services listeners running on the terminal server. This is an array of type [PLISTENERENUM](#Section_3c46804ffff447799f961ceb9a6c9559).

**Level:**  The level of information that is requested for the listeners. The only supported value is 1.

**pNumListeners:** The number of listeners returned.

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcGetSessionProtocolLastInputTime (Opnum 9)

The RpcGetSessionProtocolLastInputTime method returns the protocol status and the time the last input was received by the protocol associated with a specific session running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). The caller MUST have WINSTATION\_QUERY permission for the session. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. HRESULT RpcGetSessionProtocolLastInputTime(
2. [in] handle\_t hBinding,
3. [in] ULONG SessionId,
4. [in] PROTOCOLSTATUS\_INFO\_TYPE InfoType,
5. [out, size\_is(,\*pcbProtoStatus )]
6. unsigned char\*\* ppProtoStatus,
7. [out] ULONG\* pcbProtoStatus,
8. [out] hyper\* pLastInputTime
9. );

**hBinding:**  The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee). For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**SessionId:**  The ID of the session from which information is to be retrieved.

**InfoType:**  Specifies what type of information to gather. This is of type [PROTOCOLSTATUS\_INFO\_TYPE](#Section_dd616139394c4ac281c7175024327604).

**ppProtoStatus:**  The buffer that contains protocol status data. This data is of type PROTOCOLSTATUS, specified in section [2.2.2.20.1](#Section_c9066753acbd46789a728fb1b080bd09).

**pcbProtoStatus:**  The number of bytes of protocol data returned.

**pLastInputTime:**  The time the last input was received by the server.

Time is measured as the number of 100-nanosecond intervals since January 1, 1601 (UTC).

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion |

##### RpcGetUserCertificates (Opnum 10)

The RpcGetUserCertificates method returns a client X509 certificate if the client used the certificate to connect to a session running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). For more information, see [[X509]](https://go.microsoft.com/fwlink/?LinkId=90590). The caller MUST have WINSTATION\_QUERY permission for the session. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. HRESULT RpcGetUserCertificates(
2. [in] handle\_t hBinding,
3. [in] ULONG SessionId,
4. [out] ULONG\* pcCerts,
5. [out, size\_is(, \*pcbCerts)] byte\*\* ppbCerts,
6. [out] ULONG\* pcbCerts
7. );

**hBinding:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee). For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**SessionId:** The ID of the session for which the certificate is to be retrieved.

**pcCerts:** The number of client certificates returned.

**ppbCerts:** Certificate data.

**pcbCerts:** The size, in bytes, of *ppbCerts*.

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcQuerySessionData (Opnum 11)

The RpcQuerySessionData method returns information about a particular session running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). The caller MUST have WINSTATION\_QUERY permission to the session being queried. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. HRESULT RpcQuerySessionData(
2. [in] handle\_t hBinding,
3. [in] ULONG SessionId,
4. [in] QUERY\_SESSION\_DATA\_TYPE type,
5. [in, unique, size\_is(cbInputData )]
6. byte\* pbInputData,
7. [in, range(0, 8192)] DWORD cbInputData,
8. [out, ref, size\_is(cbSessionData), length\_is(\*pcbReturnLength)]
9. byte\* pbSessionData,
10. [in, range(0, 8192)] ULONG cbSessionData,
11. [out, ref] ULONG\* pcbReturnLength,
12. [out, ref] ULONG\* pcbRequireBufferSize
13. );

**hBinding:**  The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee). For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**SessionId:**  The ID of the session for which data is to be retrieved.

**type:**  The type of data to retrieve about the session. This is of type [QUERY\_SESSION\_DATA\_TYPE](#Section_8244f1b1ce6246f68b62949abd789b0f).

**pbInputData:**  Input data. This is a string specifying the name of the virtual channel and is required only when querying virtual channel information.

**cbInputData:**  The size, in bytes, of input data.

**pbSessionData:**  The output data containing the requested information. The data returned is of type [WDCONFIG](#Section_d1bf099beb544ed9a7230b1062dbc128) if the type specified is QUERY\_SESSION\_DATA\_WDCONFIG. It is of type [WINSTATIONVALIDATIONINFORMATION](#Section_5c439d8db58b4b0da2e5467aa05f993b) if the type specified is QUERY\_SESSION\_DATA\_LICENSE\_VALIDATION. For other types, it is protocol-specific.

**cbSessionData:**  The size, in bytes, of **pbSessionData**.

**pcbReturnLength:**  The length of the returned data, in bytes.

**pcbRequireBufferSize:**  The buffer size, in bytes, required by the returned data.

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion |

#### RCMListener

The RCMListener interface provides methods that open, close, start, and stop a [**listener**](#gt_1e5aa0c2-4617-4d7a-969e-a7091c415c8d). The version for this interface is 1.0.

For information about [**endpoints**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee), [**UUID**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) values, and versions, see sections [2.1](#Section_ac0535d0a8474386ae3ee93a50ff2484) and [1.9](#Section_4f5ff56828874c508b3638bb85170b58).

Methods in RPC Opnum Order

| Method | Description |
| --- | --- |
| [RpcOpenListener](#Section_126bc7fa9e644b7ba78c1a08895fb60a) | Returns a [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to the specified [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) listener running on a terminal server.  Opnum: 0 |
| [RpcCloseListener](#Section_95be2185992148b0b722848ae3075b9d) | Closes the handle for a Terminal Services listener running on a terminal server.  Opnum: 1 |
| [RpcStopListener](#Section_64f7fe27fea14e318495f07aae42aed1) | Stops the specified Terminal Services listener running on a terminal server.  Opnum: 2 |
| [RpcStartListener](#Section_e3aa6981665a4001b5bc996ba68c3f0e) | Starts the specified Terminal Services listener on a terminal server.  Opnum: 3 |
| [RpcIsListening](#Section_13bb2b29a4b048e0a1d2d73ea67b3069) | Checks if the specified Terminal Services listener is running on a terminal server.  Opnum: 4 |

##### RpcOpenListener (Opnum 0)

The RpcOpenListener method returns a [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to the specified [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) [**listener**](#gt_1e5aa0c2-4617-4d7a-969e-a7091c415c8d) running on a terminal server. No special permissions are required to call this method.

1. HRESULT RpcOpenListener(
2. [in] handle\_t hBinding,
3. [in, string] WCHAR\* szListenerName,
4. [out] HLISTENER\* phListener
5. );

**hBinding:**  The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle. For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**szListenerName:**  The name of the listener for which to retrieve a handle.

**phListener:**  Pointer to a handle to the listener. The handle is of type [HLISTENER](#Section_c50c3e487a1d4fb689341663a90cadff).

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcCloseListener (Opnum 1)

The RpcCloseListener method closes the [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) for a [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) [**listener**](#gt_1e5aa0c2-4617-4d7a-969e-a7091c415c8d) running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). This MUST be called after [RpcOpenListener](#Section_126bc7fa9e644b7ba78c1a08895fb60a). The call to this method MUST be serialized if there are multiple threads running otherwise the behavior of this function is unknown. No special permissions are required to call this method.

1. HRESULT RpcCloseListener(
2. [in, out] HLISTENER\* phListener
3. );

**phListener:** Pointer to a handle to the listener as returned by RpcOpenListener. The handle is of type [HLISTENER](#Section_c50c3e487a1d4fb689341663a90cadff). The handle is set to NULL when the call returns.

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.[<157>](#Appendix_A_157" \o "Product behavior note 157)

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcStopListener (Opnum 2)

The RpcStopListener method stops the specified [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) [**listener**](#gt_1e5aa0c2-4617-4d7a-969e-a7091c415c8d) running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). This MUST be called after [RpcOpenListener](#Section_126bc7fa9e644b7ba78c1a08895fb60a). The caller MUST have WINSTATION\_RESET permission to the listener. The method checks whether the caller has WINSTATION\_RESET permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. HRESULT RpcStopListener(
2. [in] HLISTENER hListener
3. );

**hListener:** The [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to the listener. This is of type [HLISTENER](#Section_c50c3e487a1d4fb689341663a90cadff).

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcStartListener (Opnum 3)

The RpcStartListener method starts the specified [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) [**listener**](#gt_1e5aa0c2-4617-4d7a-969e-a7091c415c8d) on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). This MUST be called after [RpcOpenListener](#Section_126bc7fa9e644b7ba78c1a08895fb60a). The caller MUST have WINSTATION\_RESET and WINSTATION\_QUERY permissions to the listener. The method checks whether the caller has WINSTATION\_RESET and WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permissions.

1. HRESULT RpcStartListener(
2. [in] HLISTENER hListener
3. );

**hListener:**  The [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to the listener. This is of type [HLISTENER](#Section_c50c3e487a1d4fb689341663a90cadff).

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

##### RpcIsListening (Opnum 4)

The RpcIsListening method checks whether the specified [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) [**listener**](#gt_1e5aa0c2-4617-4d7a-969e-a7091c415c8d) is running on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). This MUST be called after [RpcOpenListener](#Section_126bc7fa9e644b7ba78c1a08895fb60a). The caller MUST have WINSTATION\_QUERY permission to the listener. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. HRESULT RpcIsListening(
2. [in] HLISTENER hListener,
3. [out] BOOL\* pbIsListening
4. );

**hListener:** The [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) to the listener. This is of type [HLISTENER](#Section_c50c3e487a1d4fb689341663a90cadff).

**pbIsListening:** Set to TRUE if the listener is listening for a connection, FALSE otherwise.

**Return Values:**  The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

### Timer Events

None.

### Other Local Events

None.

## Legacy Client Details

### Abstract Data Model

None.

### Timers

No protocol timers are required beyond those used internally by [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) to implement resiliency to network outages, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15).

### Initialization

The client MUST create an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) connection to the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998), using the details specified in section [2.1](#Section_ac0535d0a8474386ae3ee93a50ff2484).

### Message Processing Events and Sequencing Rules

When a method completes, the values returned by [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) MUST be returned unmodified to the upper layer.

### Timer Events

There are no timer events.

### Other Local Events

None.

## Legacy Server Details

### Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

#### Abstract Data Types

**dwCounterID**: Identifier of a performance counter. It is used with [RpcWinStationGetTermSrvCountersValue](#Section_46f107c4f0904802b6d8c510cfcb9a93) and MUST be set to one of the values described in section [2.2.2.17.1](#Section_e08317cba69340d39a5f61e14cfdb431).

**hServer**: Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). This handle is returned by the [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a) method and is used as an input parameter with all other [LegacyApi](#Section_d9f6ec940483417695325b47455133ed) methods.

**PDCONFIG2**: Structure that contains the protocol driver's software configuration. This structure is described in detail in section [2.2.2.26.1](#Section_74204022eb7c4454b3d024f642c892a4). It is returned by the [RpcWinStationQueryInformation](#Section_1bba9ff271d349a3bb262e5f6fcab3ee) method as part of transport protocol driver configuration data.

**WinStationInformation**: Provides the current values of various properties for a session such as state, connect time, last input time, and so on. It is used with RpcWinStationQueryInformation and is returned as the *pWinStationInformation* parameter when "WinStationInformation" is passed as the *WinStationInformationClass* parameter.

**WINSTATIONPRODID**: Structure that represents a product ID for the session. This structure is described in detail in section [2.2.2.36](#Section_42e13db86eb349d3a66762500e568db5). It is used with RpcWinStationQueryInformation and is returned as the *pWinStationInformation* parameter when "WinStationDigProductId" is passed as the *WinStationInformationClass* parameter.

**WINSTATIONUSERTOKEN**: Structure that defines the user token for a session. This structure is described in detail in section [2.2.2.32](#Section_07f9831c633143e5ba27d3d58772eb4c). It is used with RpcWinStationQueryInformation and is returned as the *pWinStationInformation* parameter when "WinStationUserToken" is passed as the *WinStationInformationClass* parameter.

**POLICY\_TS\_MACHINE**: Structure that defines the machine policy of the server. This structure is described in detail in section [2.2.2.31](#Section_d7cb8c2df60e42d6b8f4e617ad4d8c1b). It is used with the [RpcWinStationGetMachinePolicy](#Section_cda4de46ea1c49aebeb0afd0f0be90f4) method and is returned as the *pPolicy* parameter.

**WINSTATIONINFOCLASS**: Enumeration that specifies the class of data to retrieve. It is defined in section [2.2.1.8](#Section_f333c223de8a46e1a83e79cbdab92371). The following list shows how **WINSTATIONINFOCLASS** is used with various methods.

* RpcWinStationQueryInformation: as the *WinStationInformationClass* parameter.
* [RpcWinStationSetInformation](#Section_2a5ee131a1dd44c7988098df708061ea): as the *WinStationInformationClass* parameter.

**USERCONFIG**: Structure that contains user and session configuration information. This structure is described in detail in section [2.2.2.18](#Section_dba750b8cb354e889811e2a1f8a10701). It is used with the RpcWinStationQueryInformation and RpcWinStationSetInformation methods.

**WINSTATIONCLIENT**: Structure that defines the client-requested configuration when connecting to a session. This structure is described in detail in section [2.2.2.19](#Section_673d8ac0f55748cb98a649925160d729). It is used with the RpcWinStationQueryInformation method.

**LOGONID**: Structure that represents information about the session or WinStation. This structure is described in detail in section [2.2.2.13](#Section_e5c755e86d66494d873f4540267fe2f4). It is used with the [RpcWinStationEnumerate](#Section_fa654a9ca67f4e589fa75dd0ae6305a1) method and is returned as the *pLogonId* parameter.

**TS\_PROCESS\_INFORMATION\_NT4**: Structure that represents information about a process running in a session. This structure is described in detail in section [2.2.2.14](#Section_0a8a73d93c9e4d2d91e47fef48b6c29d). It is used with the [RpcWinStationEnumerateProcesses](#Section_3559bc198ca34bcf842ecf25f592dc1a) method and is returned inside the *pProcessBuffer* parameter.

**TS\_ALL\_PROCESS\_INFO**: Structure that contains data about all the processes running on the system that are accessible to a user. This structure is described in detail in section [2.2.2.15](#Section_8f5564a61b5e4381afc44acf7ad39d6d). It is used with the [RpcWinStationGetAllProcesses](#Section_6f43f9e42d804c5cbc0b7f653b1d7c02) method and is returned as the *ppTsAllProcessesInfo* parameter.

**TS\_SYS\_PROCESS\_INFORMATION**: Structure that contains information about a process running in a session. This structure is described in detail in section [2.2.2.15.1](#Section_96702c7d6f5f49658f295fdbea73278e). It is used with the RpcWinStationEnumerateProcesses method and is returned inside the *pProcessBuffer* parameter.

**TS\_PROCESS\_INFORMATION\_NT6**: Structure that represents information about a process running in a session. This structure is described in detail in section [2.2.2.16](#Section_d60fc5882cc744649c53879142f5a58d). It is used with the [RpcWinStationGetAllProcesses\_NT6](#Section_e1647da01159428491bc2c2024b14e20) method and is returned inside the *ppTsAllProcessesInfo* parameter.

**PDPARAMS**: Structure containing the protocol driver parameters. This structure is described in detail in section [2.2.2.21](#Section_ff0e2998fb8b4dffab648427fad556eb). It is used with the RpcWinStationQueryInformation and RpcWinStationSetInformation methods.

**PDPARAMSWIRE**: Structure precedes a **PDPARAMS** structure and defines the size and offset of the **PDPARAMS** structures in the complete data BLOB. This structure is described in detail in section [2.2.2.46](#Section_2e867f3dfc4d4ff6a01fec5d3c1173fa). It is used with the RpcWinStationQueryInformation and RpcWinStationSetInformation methods.

**PdName**: String containing the transport protocol type. This MUST be any of the following strings: "tcp", "netbios", "ipx", or "spx". It is used with the [RpcWinStationGetLanAdapterName](#Section_f3450fddd5b54a3190893f60b6d82306) method as the *pPdName* parameter.

**WDCONFIG**: Structure containing the WinStation (session) driver configuration. This structure is described in detail in section [2.2.2.27](#Section_d1bf099beb544ed9a7230b1062dbc128). It is used with the RpcWinStationQueryInformation method.

**CDCONFIG**: Structure that represents connection driver configuration. This structure is described in detail in section [2.2.2.28](#Section_f88900e1c1594f02b3ae84f05eec212f). It is used with RpcWinStationQueryInformation and is returned as the *pWinStationInformation* parameter when "WinStationCd" is passed as the *WinStationInformationClass* parameter.

**WINSTATIONCREATE**: Structure that represents a session to which the user can connect. This structure is described in detail in section [2.2.2.29](#Section_dcd01340508440898173ad3a5831e087). It is used with RpcWinStationQueryInformation and is returned as the *pWinStationInformation* parameter when "WinStationCreateData" is passed as the *WinStationInformationClass* parameter.

**WINSTATIONVIDEODATA**: Structure that represents the resolution and color depth of a session. This structure is described in detail in section [2.2.2.33](#Section_5f95f65789d2472db4abb0595618dbd1). It is used with RpcWinStationQueryInformation and is returned as the *pWinStationInformation* parameter when "WinStationVideoData" is passed as the *WinStationInformationClass* parameter.

**WINSTATIONLOADINDICATORDATA**: Structure that represents the data used for the load balancing of a server. This structure is described in detail in section [2.2.2.34](#Section_91459fa177e84987a6f0fe7dd3e62bfc). It is used with RpcWinStationQueryInformation and is returned as the *pWinStationInformation* parameter when "WinStationLoadIndicator" is passed as the *WinStationInformationClass* parameter.

**WINSTATIONSHADOW**: Structure that represents the current shadow state of a session. This structure is described in detail in section [2.2.2.35](#Section_5ffd248b69324937b6044ca4456e8028). The following list shows how **WINSTATIONSHADOW** is used with various methods.

* RpcWinStationQueryInformation: as the *pWinStationInformation* parameter when "WinStationShadowInfo" is passed as the *WinStationInformationClass* parameter.
* RpcWinStationSetInformation: as the *pWinStationInformation* parameter when "WinStationShadowInfo" is passed as the *WinStationInformationClass* parameter.

**DEVICENAME**: Represents the name of a device. It is described in detail in section [2.2.1.7](#Section_56393cb5627846668d90be58cbec0880) and is used with RpcWinStationGetLanAdapterName.

**WINSTATIONNAME**: Represents the name of a session. It is described in detail in section [2.2.1.5](#Section_b34d2909b0cd407e934ca2c7cb96ca26) and is used with [RpcLogonIdFromWinStationName](#Section_b1397fdad4f24355a936ca44ef4dc440).

**WINSTATIONREMOTEADDRESS**: Structure that contains the client's remote address. This structure is described in detail in section [2.2.2.37](#Section_5b5a44b3155f4159aad9794e97275a3f). It is returned by the RpcWinStationQueryInformation method.

**IdleTime**: Represents the idle time for the session, in seconds. It is of type ULONG. It is used with RpcWinStationQueryInformation and is returned as the *pWinStationInformation* parameter when "WinStationIdleTime" is passed as the *WinStationInformationClass* parameter.

**LastReconnectType**: Represents the last reconnect type for the session. It is of type ULONG. It is used with RpcWinStationQueryInformation and is returned as the *pWinStationInformation* parameter when "WinStationReconnectType" is passed as the *WinStationInformationClass* parameter.

**WinStationVirtualData**: Represents client virtual data. It is of type BYTE. It is used with RpcWinStationQueryInformation and is returned as the *pWinStationInformation* parameter when "WinStationVirtualData" is passed as the *WinStationInformationClass* parameter.

**WINSTATIONCLIENTDATA**: Structure used to send data to the client. It is described in [2.2.2.41](#Section_d96b68a3c0ba47ddbd219a11d9eae598). It is used with RpcWinStationSetInformation and is returned as the *pWinStationInformation* parameter when "WinStationClientData" is passed as the *WinStationInformationClass* parameter.

**TS\_TRACE**: Structure that specifies fields used for configuring tracing operations in TS binaries if they are checked. This structure is described in detail in section [2.2.2.39](#Section_ad34957553694830a1741920a7af8d5f). It is used with the RpcWinStationSetInformation method.

**WINSTACONFIGWIRE**: Structure precedes a variable length user configuration data BLOB and defines the size and offset of the user configuration data. This structure is described in detail in section [2.2.2.47](#Section_ae5355e7c6cc48abb82ad1de8b3cd6d6). It is used with the RpcWinStationQueryInformation and RpcWinStationSetInformation methods.

### Timers

**Send Message Timeout**: The time, in seconds, that the [RpcWinStationSendMessage](#Section_a8f4bc6097484bcab231df84330b0754) method waits for the user's response to the message box displayed by that method. For more information, see description of the *Timeout* parameter to the RpcWinStationSendMessage method.

### Initialization

Parameters necessary to initialize the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) protocol are specified in section [2.1](#Section_ac0535d0a8474386ae3ee93a50ff2484).

### Message Processing Events and Sequencing Rules

This protocol asks the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) runtime to perform a strict NDR data-consistency check at target level 6.0 for all methods, unless otherwise specified, as defined in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 1.3.

When a method completes, the values returned by RPC MUST be returned unmodified to the upper layer. The methods MAY throw an exception and the [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) client MUST handle these exceptions by returning the unmodified exception code to the upper layer.

Legacy server methods are part of the [LegacyApi](#Section_d9f6ec940483417695325b47455133ed) interface.

#### LegacyApi

The LegacyApi provides legacy methods that manipulate a terminal client. The version for this interface is 1.0.

For [**endpoints**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee), [**UUID**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) values, and versions, see sections [2.1](#Section_ac0535d0a8474386ae3ee93a50ff2484) and [1.9](#Section_4f5ff56828874c508b3638bb85170b58).

Methods in RPC Opnum Order

| Method | Description |
| --- | --- |
| [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a) | Returns a server [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) that can be used in other [**WinStation**](#gt_8cc89bb9-be94-4672-8a5d-9947b91bc952) API methods for querying information on the WinStation (sessions) on the server.  Opnum: 0 |
| [RpcWinStationCloseServer](#Section_89048acf28c64d469742eef51f507f05) | Closes the server handle for WinStation APIs.  Opnum: 1 |
| [RpcIcaServerPing](#Section_fb4fa20ad77548b6b9b6edab02296225) | Verifies that the server is alive.  Opnum: 2 |
| [RpcWinStationEnumerate](#Section_fa654a9ca67f4e589fa75dd0ae6305a1) | Retrieves a list of [LOGONID](#Section_e5c755e86d66494d873f4540267fe2f4) structures for sessions on a terminal server.  Opnum: 3 |
| [RpcWinStationRename](#Section_84c75092c96a4beeac508f1632bc2f7b) | Enables the caller to change the name of the session.  Opnum: 4 |
| [RpcWinStationQueryInformation](#Section_1bba9ff271d349a3bb262e5f6fcab3ee) | Retrieves various types of configuration information on a session.  Opnum: 5 |
| [RpcWinStationSetInformation](#Section_2a5ee131a1dd44c7988098df708061ea) | Sets various types of configuration information for a session.  Opnum: 6 |
| [RpcWinStationSendMessage](#Section_a8f4bc6097484bcab231df84330b0754) | Displays a message box on a given terminal server session and, optionally, waits for a reply.  Opnum: 7 |
| [RpcLogonIdFromWinStationName](#Section_b1397fdad4f24355a936ca44ef4dc440) | Given a session name, returns the session's session ID.  Opnum: 8 |
| [RpcWinStationNameFromLogonId](#Section_d2c62b1094894e2b9dd0efd6d7850881) | Retrieves the Windows Station (WinStation) name for a specific session.  Opnum: 9 |
| [RpcWinStationConnect](#Section_88a127d180de407aa95b9fb9d45e29d4) | Connects a user's terminal server client from a given terminal server session to a different terminal server session.  Opnum: 10 |
| Opnum11NotUsedOnWire | Reserved for local use.  Opnum: 11 |
| Opnum12NotUsedOnWire | Reserved for local use.  Opnum: 12 |
| [RpcWinStationDisconnect](#Section_194a4899474546ae9697b44a5457d77d) | On the server, disconnects the terminal server client from a session.  Opnum: 13 |
| [RpcWinStationReset](#Section_2777bdbaa4c7480bae34fb234d207bd2) | Resets a session.  Opnum: 14 |
| [RpcWinStationShutdownSystem](#Section_81c472be2a71424e8bde71e826a34c26) | Shuts down the system and, optionally, logs off all sessions. May also reboot the system.  Opnum: 15 |
| [RpcWinStationWaitSystemEvent](#Section_e901b3d9eb9a43af8c929feacdf9cbc6) | Waits synchronously for a system event from an RPC API request on behalf of the caller.  Opnum: 16 |
| [RpcWinStationShadow](#Section_8115399df159477fa871522fde0cc32a) | Starts a shadow operation (remote control) of another terminal server session.  Opnum: 17 |
| Opnum18NotUsedOnWire | Reserved for local use.  Opnum: 18 |
| Opnum19NotUsedOnWire | Reserved for local use.  Opnum: 19 |
| Opnum20NotUsedOnWire | Reserved for local use.  Opnum: 20 |
| Opnum21NotUsedOnWire | Reserved for local use.  Opnum: 21 |
| Opnum22NotUsedOnWire | Reserved for local use.  Opnum: 22 |
| Opnum23NotUsedOnWire | Reserved for local use.  Not implemented.  Opnum: 23 |
| Opnum24NotUsedOnWire | Reserved for local use.  Opnum: 24 |
| Opnum25NotUsedOnWire | Reserved for local use.  Opnum: 25 |
| Opnum26NotUsedOnWire | Reserved for local use.  Opnum: 26 |
| Opnum27NotUsedOnWire | Reserved for local use.  Opnum: 27 |
| Opnum28NotUsedOnWire | Reserved for local use.  Opnum: 28 |
| [RpcWinStationBreakPoint](#Section_5b55d7a032c5494ba1a91d25d283c542) | Breaks into the debugger in either the session process of a specific session or in the terminal server service process.  Opnum: 29 |
| [RpcWinStationReadRegistry](#Section_5d74a6b079fd4091a2eb4974131b4ea4) | Tells the server to reread from the registry the configuration data for all the WinStations.  Opnum: 30 |
| Opnum31NotUsedOnWire | Reserved for local use.  Opnum: 31 |
| Opnum32NotUsedOnWire | Reserved for local use.  Opnum: 32 |
| Opnum33NotUsedOnWire | Reserved for local use.  Opnum: 33 |
| [OldRpcWinStationEnumerateProcesses](#Section_f2b9af5b53494f4f9d8416e5271a04d9) | This function in turn calls the [RpcWinStationEnumerateProcesses](#Section_3559bc198ca34bcf842ecf25f592dc1a) function.  Opnum: 34 |
| Opnum35NotUsedOnWire | Reserved for local use.  Opnum: 35 |
| RpcWinStationEnumerateProcesses | Returns the process information for an NT4 terminal server. Supported only for backward compatibility with that platform.  Opnum: 36 |
| [RpcWinStationTerminateProcess](#Section_60d456d751d9432199b0e22852573810) | Terminates the specified process.  Opnum: 37 |
| Opnum38NotUsedOnWire | Reserved for local use.  Opnum: 38 |
| Opnum39NotUsedOnWire | Reserved for local use.  Opnum: 39 |
| Opnum40NotUsedOnWire | Not implemented.  Opnum: 40 |
| Opnum41NotUsedOnWire | Not implemented.  Opnum: 41 |
| Opnum42NotUsedOnWire | Reserved for local use.  Opnum: 42 |
| [RpcWinStationGetAllProcesses](#Section_6f43f9e42d804c5cbc0b7f653b1d7c02) | Retrieves a list of the processes on a remote server on which the caller has permission to receive information.  Opnum: 43 |
| [RpcWinStationGetProcessSid](#Section_94fc771f75834a4d8579efac05820551) | Retrieves the process SID for a given process ID and process start time combination.  Opnum: 44 |
| [RpcWinStationGetTermSrvCountersValue](#Section_46f107c4f0904802b6d8c510cfcb9a93) | Retrieves the current value of requested terminal server performance counters.  Opnum: 45 |
| [RpcWinStationReInitializeSecurity](#Section_563daea7578644f3bf332d24e058856a) | Reinitializes security for all non-console WinStation remote connection protocols specified in the registry.  Opnum: 46 |
| Opnum47NotUsedOnWire | Reserved for local use.  Opnum: 47 |
| Opnum48NotUsedOnWire | Reserved for local use.  Opnum: 48 |
| Opnum49NotUsedOnWire | Reserved for local use.  Opnum: 49 |
| Opnum50NotUsedOnWire | Reserved for local use.  Opnum: 50 |
| Opnum51NotUsedOnWire | Reserved for local use.  Opnum: 51 |
| Opnum52NotUsedOnWire | Not implemented.  Opnum: 52 |
| [RpcWinStationGetLanAdapterName](#Section_f3450fddd5b54a3190893f60b6d82306) | Returns the LAN adapter GUID as a string.  Opnum: 53 |
| Opnum54NotUsedOnWire | Reserved for local use.  Opnum: 54 |
| Opnum55NotUsedOnWire | Reserved for local use.  Opnum: 55 |
| Opnum56NotUsedOnWire | Reserved for local use.  Opnum: 56 |
| Opnum57NotUsedOnWire | Reserved for local use.  Opnum: 57 |
| [RpcWinStationUpdateSettings](#Section_ff68952dedfe4f81aaf2b2e37d2b29ef) | Rereads settings for all WinStations.  Opnum: 58 |
| [RpcWinStationShadowStop](#Section_23f97d498b344ee68af087114260e2bb) | Stops all shadow operations on the specified session.  Opnum: 59 |
| [RpcWinStationCloseServerEx](#Section_9d198d98b7f1475d8c29cc188f5bc0fc) | Closes the server handle for WinStation APIs.  Opnum: 60 |
| [RpcWinStationIsHelpAssistantSession](#Section_113108ba2939435ea14fd45414b8ad91) | Determines whether a session is created by a HelpAssistant account (used for Remote Assistance).  Opnum: 61 |
| [RpcWinStationGetMachinePolicy](#Section_cda4de46ea1c49aebeb0afd0f0be90f4) | Returns a copy of the terminal server machine policy to the caller.  Opnum: 62 |
| Opnum63NotUsedOnWire | Reserved for local use.  Opnum: 63 |
| Opnum64NotUsedOnWire | Reserved for local use.  Opnum: 64 |
| [RpcWinStationCheckLoopBack](#Section_e8cb245a90a3424db9bdacec6e01d86a) | Checks if there is a loopback when a client tries to connect.  Opnum: 65 |
| [RpcConnectCallback](#Section_93794f61e93046f0a2a63f84531546ea) | Initiates a connection back to the Remote Assistance (RA) client.  Opnum: 66 |
| Opnum67NotUsedOnWire | Reserved for local use.  Opnum: 67 |
| Opnum68NotUsedOnWire | Reserved for local use.  Opnum: 68 |
| Opnum69NotUsedOnWire | Reserved for local use.  Opnum: 69 |
| [RpcWinStationGetAllProcesses\_NT6](#Section_e1647da01159428491bc2c2024b14e20) | Retrieves the processes running a remote server on which the caller has access to retrieve information.  Opnum: 70 |
| Opnum71NotUsedOnWire | Reserved for local use.  Opnum: 71 |
| Opnum72NotUsedOnWire | Reserved for local use.  Opnum: 72 |
| Opnum73NotUsedOnWire | Reserved for local use.  Opnum: 73 |
| Opnum74NotUsedOnWire | Reserved for local use.  Opnum: 74 |
| [RpcWinStationOpenSessionDirectory](#Section_8bd6683781b74da0809ef797bd3a5b41) | Pings the Session Directory to see if it can accept RPC calls.  Opnum: 75 |

In the preceding table, the phrase "Reserved for local use" means that the client MUST NOT send the opnum, and the server behavior is undefined, because it does not affect interoperability.[<158>](#Appendix_A_158" \o "Product behavior note 158)

##### RpcWinStationOpenServer (Opnum 0)

The RpcWinStationOpenServer method returns a server [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) that can be used in other [**WinStation**](#gt_8cc89bb9-be94-4672-8a5d-9947b91bc952) API methods for querying information about the WinStation (sessions) on the server. No special permissions are required to call this method.

1. BOOLEAN RpcWinStationOpenServer(
2. [in] handle\_t hBinding,
3. [out] DWORD\* pResult,
4. [out] SERVER\_HANDLE\* phServer
5. );

**hBinding:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle.

**pResult:**  Failure error code if the call to RpcWinStationOpenServer failed. If the call was successful, this parameter is STATUS\_SUCCESS (0x00000000) (as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90)).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | Successful call. |
| STATUS\_CANCELLED  0xC0000120 | The server is shutting down. |
| STATUS\_NO\_MEMORY  0xC0000017 | Not enough memory to complete the operation. |

**phServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). This handle is used by other RpcWinStation methods.

**Return Values:** Returns TRUE if the call succeeded, or FALSE if the call failed. On failure, pResult indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationCloseServer (Opnum 1)

The RpcWinStationCloseServer method closes the server [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) for [**WinStation**](#gt_8cc89bb9-be94-4672-8a5d-9947b91bc952) APIs. No special permissions are required to call this method.

1. BOOLEAN RpcWinStationCloseServer(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult
4. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The *hServer* MUST be returned from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:** If the call was successful, this parameter is STATUS\_SUCCESS (0x00000000), as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90); otherwise, it MUST be an implementation-specific negative value.

**Return Values:**  Returns TRUE if the call succeeded, or FALSE if the call failed. On failure, pResult indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcIcaServerPing (Opnum 2)

The RpcIcaServerPing method is called to verify that the server is alive. No special permissions are required to call this method.[<159>](#Appendix_A_159" \o "Product behavior note 159)

1. BOOLEAN RpcIcaServerPing(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult
4. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The *hServer* argument MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:** If the call was successful, this parameter is STATUS\_SUCCESS (0x00000000), as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90); otherwise it MUST be an implementation-specific negative value.

**Return Values:**  Returns TRUE if the call succeeded and the server is alive, or FALSE if the method failed. On failure, pResult indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationEnumerate (Opnum 3)

The RpcWinStationEnumerate method retrieves a list of [LOGONID](#Section_e5c755e86d66494d873f4540267fe2f4) structures for sessions on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). No special permissions are required to call this method. However, only sessions to which the caller has WINSTATION\_QUERY permission are enumerated. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. BOOLEAN RpcWinStationEnumerate(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in, out] PULONG pEntries,
5. [in, out, unique, size\_is(\*pByteCount)]
6. PCHAR pLogonId,
7. [in, out] PULONG pByteCount,
8. [in, out] PULONG pIndex
9. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The hServer argument MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:** If the call was successful, this parameter will be STATUS\_SUCCESS (0x00000000), as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90); otherwise, it MUST be an implementation-specific negative value.

**pEntries:**  Pointer to the number of entries to return to the caller. On return from this method, this is the number of logon IDs actually returned in this call to RpcWinStationEnumerate.

**pLogonId:**  Buffer where the logon IDs are stored when the method returns. This will be an array of LOGONID structures. Caller MUST cast this to PCHAR before calling this method.

**pByteCount:**  Size of the buffer, in bytes, to which *pLogonId* points.

**pIndex:**  Last index of the logon ID lookup from this call, passed to the server the next time this method is called. Initial value of this passed by the caller MUST be 0.

**Return Values:**  Returns TRUE if the call succeeded, or FALSE if the lookup failed. On failure, *pResult* indicates the failure status code. If all of the logon IDs have already been retrieved from the server, TRUE will be returned, and pResult will be STATUS\_NO\_MORE\_ENTRIES (as specified in [MS-ERREF]), indicating to the call that all logon IDs have been retrieved.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationRename (Opnum 4)

The RpcWinStationRename method enables the caller to change the name of the session. The caller MUST have DELETE permission, as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.3, on the session that is identified by the old name.[<160>](#Appendix_A_160" \o "Product behavior note 160)

1. BOOLEAN RpcWinStationRename(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in, size\_is(NameOldSize)] PWCHAR pWinStationNameOld,
5. [in, range(0, 256)] DWORD NameOldSize,
6. [in, size\_is(NameNewSize)] PWCHAR pWinStationNameNew,
7. [in, range(0, 256)] DWORD NameNewSize
8. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The *hServer* MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:**  The failure error code if the call to RpcWinStationRename fails. If the call is successful, this parameter MUST be STATUS\_SUCCESS (0x00000000), as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | The call is successful. |
| STATUS\_ACCESS\_DENIED  0xC0000022 | The caller does not have DELETE permission. |
| STATUS\_CTX\_WINSTATION\_NAME\_INVALID  0xC00A0001 | The sizes are 0, one or the other of the pointers is NULL, or a pointer is invalid. |
| STATUS\_CTX\_WINSTATION\_NOT\_FOUND  0xC00A0015 | No session exists with the name given in *pWinStationNameOld*. |
| STATUS\_CTX\_WINSTATION\_NAME\_COLLISION  0xC00A0016 | A session already exists with the name given in *pWinStationNameNew*. |

**pWinStationNameOld:**  The pointer to a string that is the old name of the session being renamed.

**NameOldSize:**  The length of the string in characters pointed to by *pWinStationNameOld* including the terminating NULL character.

**pWinStationNameNew:**  The pointer to a string that is the new name of the session being renamed.

**NameNewSize:** The length of the string in characters pointed to by *pWinStationNameNew* including the terminating NULL character. Name MUST be shorter than or equal to [WINSTATIONNAME\_LENGTH](#Section_ce70794f213843e8bf6c2c147887d6a2).

**Return Values:**  Returns TRUE if the call succeeded, or FALSE if the method failed. On failure, pResult indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationQueryInformation (Opnum 5)

The RpcWinStationQueryInformation method retrieves various types of configuration information on a session. The caller MUST have the WINSTATION\_QUERY permission right as well as specific permission rights for some operations as indicated in the following sections. The method checks whether the caller has WINSTATION\_QUERY permission and the specific permission required for some operations (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.[<161>](#Appendix_A_161" \o "Product behavior note 161)

1. BOOLEAN RpcWinStationQueryInformation(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in] DWORD LogonId,
5. [in] DWORD WinStationInformationClass,
6. [in, out, unique, size\_is(WinStationInformationLength)]
7. PCHAR pWinStationInformation,
8. [in, range(0, 0x8000)] DWORD WinStationInformationLength,
9. [out] DWORD\* pReturnLength
10. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). *hServer* MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:** Failure error code if the call to RpcWinStationQueryInformation failed. If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | Successful completion. |
| STATUS\_INVALID\_INFO\_CLASS  0xC0000003 | The class is not recognized. |
| STATUS\_BUFFER\_TOO\_SMALL  0xC0000023 | *WinStationInformationLength* is too small. |
| STATUS\_ACCESS\_DENIED  0xC0000022 | The caller does not have permission for the operation. |

**LogonId:** The session ID of the session for which to retrieve information.

**WinStationInformationClass:** The class of data to retrieve. These values come from the enum type [WINSTATIONINFOCLASS](#Section_f333c223de8a46e1a83e79cbdab92371).

The following classes are supported.

| Value | Meaning |
| --- | --- |
| WinStationCreateData  0 | Retrieves general information on the type of [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) session (protocol) to which the session belongs.  The pWinStationInformation argument points to a [WINSTATIONCREATE](#Section_dcd01340508440898173ad3a5831e087) structure, and *WinStationInformationLength* MUST be sizeof(WINSTATIONCREATE).[<162>](#Appendix_A_162" \o "Product behavior note 162) |
| WinStationConfiguration  1 | Retrieves general configuration data on the terminal server session.  The *pWinStationInformation* argument points to a [WINSTACONFIGWIRE](#Section_ae5355e7c6cc48abb82ad1de8b3cd6d6) structure followed by a [USERCONFIG](#Section_dba750b8cb354e889811e2a1f8a10701) structure. The *WinStationInformationLength* MUST be sizeof(WINSTACONFIGWIRE) + sizeof(USERCONFIG). The **Size** field in the USERCONFIG structure inside WINSTACONFIGWIRE MUST be set to sizeof(USERCONFIG) and the Offset set to sizeof(WINSTACONFIGWIRE). The **Size** field in the **NewFields** structure inside WINSTACONFIGWIRE MUST be set to 0, and the offset MUST be set to sizeof(WINSTACONFIGWIRE) + sizeof(USERCONFIG). |
| WinStationPdParams  2 | Retrieves transport protocol driver parameters.[<163>](#Appendix_A_163" \o "Product behavior note 163) The structure coming into the function indicates via SDClass the specific protocol driver on which to receive parameter information. The result will be returned in the union in the structure.  The *pWinStationInformation* argument points to a [PDPARAMSWIRE](#Section_2e867f3dfc4d4ff6a01fec5d3c1173fa) structure followed by a [PDPARAMS](#Section_ff0e2998fb8b4dffab648427fad556eb) structure. The **WinStationInformationLength** MUST be sizeof(PDPARAMSWIRE) + sizeof(PDPARAMS). The **Size** field in **SdClassSpecific** field inside PDPARAMSWIRE MUST be set to sizeof(PDPARAMS) and the offset MUST be set to sizeof(PDPARAMSWIRE). |
| WinStationWd  3 | Retrieves WinStation protocol driver configuration data for the session.[<164>](#Appendix_A_164" \o "Product behavior note 164)  The *pWinStationInformation* argument points to a [VARDATA\_WIRE](#Section_cc15eeb5f98148d193f6d4f9bfdb07ca) structure followed by a [WDCONFIG](#Section_d1bf099beb544ed9a7230b1062dbc128) structure. The **WinStationInformationLength** MUST be sizeof(VARDATA\_WIRE) + sizeof(WDCONFIG). The Size field in the VARDATA\_WIRE structure MUST be set to sizeof(WDCONFIG) and the Offset set to sizeof(VARDATA\_WIRE). |
| WinStationPd  4 | Retrieves transport protocol driver configuration data for the session.[<165>](#Appendix_A_165" \o "Product behavior note 165)  The *pWinStationInformation* argument points to a VARDATA\_WIRE, followed by a PDPARAMSWIRE structure, followed by a [PDCONFIG2](#Section_74204022eb7c4454b3d024f642c892a4) structure and finally ending with a PDPARAMS structure. The **WinStationInformationLength** MUST be sizeof(VARDATA\_WIRE) + sizeof(PDPARAMSWIRE) + size of(PDCONFIG2) + sizeof(PDPARAMS). The Size field in the VARDATA\_WIRE structure MUST be set to sizeof(PDCONFIG2) and the Offset set to sizeof(VARDATA\_WIRE) + sizeof(PDPARAMSWIRE). The Size field in SdClassSpecific field inside PDPARAMSWIRE MUST be set to sizeof(PDPARAMS) - sizeof([SDCLASS](#Section_7de560c9ca6b4741a448c67de0e52e0e)), and the offset MUST be set to Offset + Size of the VARDATA\_WIRE structure. |
| WinStationPrinter  5 | Not supported. |
| WinStationClient  6 | Retrieves data on the terminal server client of the session.  The *pWinStationInformation* argument points to a VARDATA\_WIRE structure followed by a [WINSTATIONCLIENT](#Section_673d8ac0f55748cb98a649925160d729) structure. The *WinStationInformationLength* MUST be sizeof(VARDATA\_WIRE) + sizeof(WINSTATIONCLIENT). The Size field in the VARDATA\_WIRE structure MUST be set to sizeof(WINSTATIONCLIENT) and the Offset set to sizeof(VARDATA\_WIRE). |
| WinStationModules  7 | Internal get function to retrieve data on protocol-specific binaries loaded for the given terminal server session. The structure pointed to by *pWinStationInformation* and the size of the buffer is Terminal Service protocol-specific. |
| WinStationInformation  8 | Retrieves information on the session, including connect state, session's name, connect time, disconnect time, time last input was received from the client, logon time, user's username and [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca), and the current time.  *pWinStationInformation* points to a VARDATA\_WIRE structure followed by a [WINSTATIONINFORMATION](#Section_c2566d8b7016440bb7e00d07c3b2418f) structure. The *WinStationInformationLength* MUST be sizeof(VARDATA\_WIRE) + sizeof(WINSTATIONINFORMATION). The **Size** field in the VARDATA\_WIRE structure MUST be set to sizeof(WINSTATIONINFORMATION) and the **Offset** set to sizeof(VARDATA\_WIRE). |
| WinStationUserToken  14 | Retrieves the user's token in the session. Caller requires WINSTATION\_ALL\_ACCESS permission.  The *pWinStationInformation* argument points to a [WINSTATIONUSERTOKEN](#Section_07f9831c633143e5ba27d3d58772eb4c) structure, and *WinStationInformationLength* MUST be sizeof(WINSTATIONUSERTOKEN). |
| WinStationVideoData  16 | Retrieves resolution and color depth of the session.  The *pWinStationInformation* argument points to a [WINSTATIONVIDEODATA](#Section_5f95f65789d2472db4abb0595618dbd1) structure, and *WinStationInformationLength* MUST be sizeof(WINSTATIONVIDEODATA). |
| WinStationCd  18 | Retrieves connection driver configuration data.  The *pWinStationInformation* points to a [CDCONFIG](#Section_f88900e1c1594f02b3ae84f05eec212f) structure, and *WinStationInformationLength* MUST be sizeof(CDCONFIG). |
| WinStationVirtualData  20 | Query client virtual data.  The *pWinStationInformation* argument MUST point to 8 bytes and *WinStationInformationLength* MUST be 8. |
| WinStationLoadBalanceSessionTarget  24 | Retrieves the target session ID for a client redirected from another server in a load balancing cluster.  The *pWinStationInformation* points to a ULONG, and *WinStationInformationLength* MUST be sizeof(ULONG). If there is no redirection, -1 is returned in *pWinStationInformation*.[<166>](#Appendix_A_166" \o "Product behavior note 166) |
| WinStationLoadIndicator  25 | Retrieves an indicator of the load on the server.  The *pWinStationInformation* argument points to a [WINSTATIONLOADINDICATORDATA](#Section_91459fa177e84987a6f0fe7dd3e62bfc) structure. *WinStationInformationLength* MUST be sizeof(WINSTATIONLOADINDICATORDATA). |
| WinStationShadowInfo  26 | Retrieves the current shadow state of a session.  The *pWinStationInformation* argument points to a [WINSTATIONSHADOW](#Section_5ffd248b69324937b6044ca4456e8028) structure. *WinStationInformationLength* MUST be sizeof(WINSTATIONSHADOW).[<167>](#Appendix_A_167" \o "Product behavior note 167) |
| WinStationDigProductId  27 | Retrieves the client product ID and current product ID of the session.  The *pWinStationInformation* argument points to a [WINSTATIONPRODID](#Section_42e13db86eb349d3a66762500e568db5) structure. *WinStationInformationLength* MUST be sizeof(WINSTATIONPRODID).[<168>](#Appendix_A_168" \o "Product behavior note 168) |
| WinStationLockedState  28 | Retrieves the current locked state of the session, TRUE or FALSE.  The *pWinStationInformation* argument points to a BOOL variable. *WinStationInformationLength* MUST be sizeof(BOOL). |
| WinStationRemoteAddress  29 | Retrieves the remote IP address of the terminal server client in the session.  The *pWinStationInformation* argument points to a [WINSTATIONREMOTEADDRESS](#Section_5b5a44b3155f4159aad9794e97275a3f) structure. *WinStationInformationLength* MUST be sizeof(WINSTATIONREMOTEADDRESS). |
| WinStationIdleTime  30 | Retrieves the idle time for the session, in seconds.  The *pWinStationInformation* argument points to a ULONG variable.  *WinStationInformationLength* MUST be sizeof(ULONG).[<169>](#Appendix_A_169" \o "Product behavior note 169) |
| WinStationLastReconnectType  31 | Retrieves the last reconnect type for the session. The value placed in *pWinStationInformation* will come from the enum [RECONNECT\_TYPE](#Section_f368f3a624824943b034bd2753de12e7).  The *pWinStationInformation* argument points to a ULONG variable.  *WinStationInformationLength* MUST be sizeof(ULONG).[<170>](#Appendix_A_170" \o "Product behavior note 170) |
| WinStationDisallowAutoReconnect  32 | Retrieves the allow (1) or disallow (0) state for auto-reconnect, 1 or 0.  The *pWinStationInformation* argument points to a BOOLEAN variable.  *WinStationInformationLength* MUST be sizeof(BOOLEAN). |
| WinStationReconnectedFromId  37 | In case of reconnected sessions, this will return the session ID of the temporary session from which it was reconnected, or -1 if no temporary session was created.  The *WinStationInformationLength* argument points to a ULONG variable.  *WinStationInformationLength* MUST be sizeof(ULONG). |
| WinStationEffectsPolicy  38 | Return policies that differentiate among implementations.  The *pWinStationInformation* argument points to a ULONG variable.  *WinStationInformationLength* MUST be sizeof(ULONG). |
| WinStationType  39 | Returns the type associated with this WinStation.  The *pWinStationInformation* argument points to a ULONG variable.  *WinStationInformationLength* MUST be sizeof(ULONG). |
| WinStationInformationEx  40 | Retrieves extended information on the session, including connect state, flags, session's name, connect time, disconnect time, time last input was received from the client, logon time, user's username and domain, and the current time.  *pWinStationInformation* points to a VARDATA\_WIRE structure followed by a WINSTATIONINFORMATIONEX structure.  The *WinStationInformationLength* MUST be sizeof(VARDATA\_WIRE) + sizeof(WINSTATIONINFORMATIONEX). The Size field in the VARDATA\_WIRE structure MUST be set to sizeof(WINSTATIONINFORMATIONEX) and the Offset set to sizeof(VARDATA\_WIRE). |

**pWinStationInformation:** Pointer to buffer allocated by the caller in which to retrieve the data. The data type or structure that *pWinStationInformation* points to is determined by the value of *WinStationInformationClass*. See previous sections for what *pWinStationInformation* SHOULD point to for each class.

**WinStationInformationLength:** Size of the data pointed to by *pWinStationInformation*, in bytes.

**pReturnLength:** Pointer to a variable to receive the size, in bytes, of the data retrieved. If *WinStationInformationLength* is too small, *pReturnLength* indicates the correct number of bytes for the caller to allocate.

**Return Values:**  Returns TRUE if the call succeeded, or FALSE if the method failed. On failure, *pResult* indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationSetInformation (Opnum 6)

The RpcWinStationSetInformation method sets various types of configuration information for a session. The caller MUST have the WINSTATION\_SET permission. Some operations MUST have more specific permissions as indicated in more detail in the sections that follow. The method checks whether the caller has WINSTATION\_SET permission and the specific permission for the configuration information (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permissions.[<171>](#Appendix_A_171" \o "Product behavior note 171)

1. BOOLEAN RpcWinStationSetInformation(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in] DWORD LogonId,
5. [in] DWORD WinStationInformationClass,
6. [in, out, unique, size\_is(WinStationInformationLength)]
7. PCHAR pWinStationInformation,
8. [in, range(0, 0x8000)] DWORD WinStationInformationLength
9. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). *hServer* MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:** Failure error code if the call to RpcWinStationSetInformation failed. If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | Successful completion. |
| STATUS\_INVALID\_INFO\_CLASS  0xC0000003 | The class is not recognized. |
| STATUS\_ACCESS\_DENIED  0xC0000022 | The caller does not have permission for the operation. |

**LogonId:**  The ID of the session for which to set information.

**WinStationInformationClass:**  The class of data to set. These values come from the enum type [WINSTATIONINFOCLASS](#Section_f333c223de8a46e1a83e79cbdab92371). See the following sections for the supported classes.

| Value | Meaning |
| --- | --- |
| WinStationConfiguration  1 | Merges configuration data into the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) session's data.  The *pWinStationInformation* argument points to a [WINSTACONFIGWIRE](#Section_ae5355e7c6cc48abb82ad1de8b3cd6d6) structure followed by a [USERCONFIG](#Section_dba750b8cb354e889811e2a1f8a10701) structure. The *WinStationInformationLength* MUST be sizeof(WINSTACONFIGWIRE) + sizeof(USERCONFIG). The **Size** field in the USERCONFIG structure inside WINSTACONFIGWIRE MUST be set to sizeof(USERCONFIG) and the Offset set to sizeof(WINSTACONFIGWIRE). The **Size** field in the **NewFields** structure inside WINSTACONFIGWIRE MUST be set to 0, and the offset MUST be set to sizeof(WINSTACONFIGWIRE) + sizeof(USERCONFIG). |
| WinStationPdParams  2 | Sets transport protocol driver parameters. The structure coming into the function indicates via SDClass the specific protocol driver for which to set the parameter information.  The *pWinStationInformation* argument points to a [PDPARAMSWIRE](#Section_2e867f3dfc4d4ff6a01fec5d3c1173fa) structure followed by a [PDPARAMS](#Section_ff0e2998fb8b4dffab648427fad556eb) structure. The **WinStationInformationLength** MUST be sizeof(PDPARAMSWIRE) + sizeof(PDPARAMS). The **Size** field in **SdClassSpecific** field inside PDPARAMSWIRE MUST be set to sizeof(PDPARAMS) and the offset MUST be set to sizeof(PDPARAMSWIRE). |
| WinStationTrace  9 | Enables tracing on the lower-level terminal server drivers for this session. This MUST be called by a process running as SYSTEM or as an [**administrator**](#gt_eb335f5b-619b-46ed-9138-06e9910108c3). The method performs access checks as defined in sections [3.1.2](#Section_44ae75e5641442258d5331426d7990a8) and [3.1.3](#Section_d8999c6c1ce34e9cb24d464651a2904b). The method fails if both checks fail.  The *pWinStationInformation* argument points to a [TS\_TRACE](#Section_ad34957553694830a1741920a7af8d5f) structure, and *WinStationInformationLength* MUST be sizeof(TS\_TRACE). |
| WinStationBeep  10 | Sends a beep to the session.  The *pWinStationInformation* argument points to a [BEEPINPUT](#Section_2faf4c5b3a79491a9d1a145f46a797d1) structure, and *WinStationInformationLength* MUST be sizeof(BEEPINPUT). |
| WinStationEncryptionOff  11 | Turns encryption off.[<172>](#Appendix_A_172" \o "Product behavior note 172) |
| WinStationEncryptionPerm  12 | Turns encryption permanently on.[<173>](#Appendix_A_173" \o "Product behavior note 173) |
| WinStationNtSecurity  13 | Sends logon service in the session a CTRL+ALT+DEL message.[<174>](#Appendix_A_174" \o "Product behavior note 174)  The *pWinStationInformation* argument and *WinStationInformationLength* are not used for this class. Set them to dummy valid data, however, as there always has to be something in these parameters. |
| WinStationInitialProgram  17 | Not used. |
| WinStationSystemTrace  19 | Enables global tracing on the lower-level terminal server drivers for this session. This MUST be called by a process running as SYSTEM or as an administrator. The method performs access checks as defined in sections 3.1.2 and 3.1.3. The method fails if both checks fail.  The *pWinStationInformation* argument points to a TS\_TRACE structure, and *WinStationInformationLength* MUST be sizeof(TS\_TRACE). |
| WinStationClientData  21 | Sends data to the terminal server client.  *WinStationInformationLength* represents the complete length of all items to send and MUST be at least sizeof([VARDATA\_WIRE](#Section_cc15eeb5f98148d193f6d4f9bfdb07ca)) + sizeof([WINSTATIONCLIENTDATA](#Section_d96b68a3c0ba47ddbd219a11d9eae598) ). Otherwise, STATUS\_INFO\_LENGTH\_MISMATCH (as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90)) is returned in *pResult*. If the data is larger than this but still less than what is expected, STATUS\_INVALID\_USER\_BUFFER (as specified in [MS-ERREF]) is returned in *pResult*.  The *pWinStationInformation* points to a VARDATA\_WIRE structure followed by the WINSTATIONCLIENTDATA structure itself. The **Size** field in the VARDATA\_WIRE structure MUST be set to sizeof(WINSTATIONCLIENTDATA) and the **Offset** set to sizeof(VARDATA\_WIRE). |
| WinStationSecureDesktopEnter  22 | Not used. |
| WinStationSecureDesktopExit  23 | Not used. |
| WinStationShadowInfo  26 | Not used. |
| WinStationLockedState  28 | Notifies processes of the new locked state of the session. TRUE or FALSE. The *pWinStationInformation* argument points to a BOOL variable. *WinStationInformationLength* MUST be sizeof (BOOL). |
| WinStationDisallowAutoReconnect  32 | Allows or disallows auto-reconnect behavior for this session, TRUE or FALSE. This MUST be called by a process running as SYSTEM.  The *pWinStationInformation* argument points to a BOOL variable. *WinStationInformationLength* MUST be sizeof (BOOL). |

**pWinStationInformation:** Pointer to buffer allocated by the caller in which the data for the operation is located. The data type or structure to which *pWinStationInformation* points is determined by the value of *WinStationInformationClass*.

**WinStationInformationLength:** Size of the data pointed to by *pWinStationInformation*, in bytes.

**Return Values:**  Returns TRUE if the call succeeded, or FALSE if the method failed. On failure, *pResult* indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationSendMessage (Opnum 7)

The RpcWinStationSendMessage method displays a message box on a given [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) session and, optionally, waits for a reply. The caller MUST have WINSTATION\_MSG permission for this method to succeed. The method checks whether the caller has WINSTATION\_MSG permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.[<175>](#Appendix_A_175" \o "Product behavior note 175)

1. BOOLEAN RpcWinStationSendMessage(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in] DWORD LogonId,
5. [in, size\_is(TitleLength)] PWCHAR pTitle,
6. [in, range(0, 1024 )] DWORD TitleLength,
7. [in, size\_is(MessageLength)] PWCHAR pMessage,
8. [in, range(0, 1024 )] DWORD MessageLength,
9. [in] DWORD Style,
10. [in] DWORD Timeout,
11. [out] DWORD\* pResponse,
12. [in] BOOLEAN DoNotWait
13. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The *hServer* argument MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:**  Failure error code if the call to RpcWinStationSendMessage failed. If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | The call succeeded. |
| STATUS\_ACCESS\_DENIED  0xC0000022 | The caller does not have WINSTATION\_MSG permission. |

**LogonId:**  The session ID of the session on which to display the message box.

**pTitle:**  Pointer to the title for the message box to display.

**TitleLength:**  The length, in bytes, of the title to display.

**pMessage:**  Pointer to the message to display.

**MessageLength:**  The length, in bytes, of the message to display in the specified session.

**Style:**  Can be any value that the standard MessageBox() method's *Style* parameter takes. For more information, see [[MSDN-MSGBOX]](https://go.microsoft.com/fwlink/?LinkId=93359).

**Timeout:**  The response time-out, in seconds. If the message box is not responded to in *Timeout* seconds, a response code of IDTIMEOUT MUST be returned in *pResponse* to indicate that the message box timed out. This time-out value is managed by another system component which dismisses the message box if no user input is entered during this interval.

**pResponse:**  The return code from the MessageBox method. This value will be a standard MessageBox return value. For more information, see [MSDN-MSGBOX].

**DoNotWait:**  If set to TRUE, do not wait for the response to the message. On return, if no errors occur in queuing the message, the *pResponse* parameter will be set to IDASYNC.

If FALSE, wait for a response.

**Return Values:**  Returns TRUE if the call succeeded, or FALSE if the method failed. On failure, *pResult* indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcLogonIdFromWinStationName (Opnum 8)

The RpcLogonIdFromWinStationName method returns a session's session ID given its session name. The caller MUST have WINSTATION\_QUERY permission. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.[<176>](#Appendix_A_176" \o "Product behavior note 176)

1. BOOLEAN RpcLogonIdFromWinStationName(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in, size\_is(NameSize)] PWCHAR pWinStationName,
5. [in, range(0, 256 )] DWORD NameSize,
6. [out] DWORD\* pLogonId
7. );

**hServer:** The Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). hServer MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:**  The failure error code if the call to RpcLogonIdFromWinStationName fails. If the call is successful, this parameter MUST be STATUS\_SUCCESS (0x00000000).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | The call is successful. |
| STATUS\_ACCESS\_DENIED  0xC0000022 | The caller does not have permission for the operation. |

**pWinStationName:**  The pointer to a buffer holding the session name.

**NameSize:**  The length of the string in characters pointed to by *pWinStationName* including the terminating NULL character. MUST be less than or equal to [WINSTATIONNAME\_LENGTH](#Section_ce70794f213843e8bf6c2c147887d6a2).

**pLogonId:** The matching session ID for the session specified by *pWinStationName*.

**Return Values:**  Returns TRUE if the call succeeded, or FALSE if the method failed. On failure, *pResult* indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationNameFromLogonId (Opnum 9)

The RpcWinStationNameFromLogonId method retrieves the [**Windows Station (WinStation)**](#gt_8cc89bb9-be94-4672-8a5d-9947b91bc952) name for a specific session. The caller MUST have WINSTATION\_QUERY permission. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.[<177>](#Appendix_A_177" \o "Product behavior note 177)

1. BOOLEAN RpcWinStationNameFromLogonId(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in] DWORD LoginId,
5. [in, out, size\_is(NameSize)] PWCHAR pWinStationName,
6. [in, range(0, 256 )] DWORD NameSize
7. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The *hServer* MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:**  Failure error code if the call to RpcWinStationNameFromLogonId failed. If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | The call succeeded. |
| STATUS\_INVALID\_PARAMETER  0xC000000D | *NameSize* value is less than [WINSTATIONNAME\_LENGTH](#Section_ce70794f213843e8bf6c2c147887d6a2) + 1. |
| 0x80071B6E | The session does not exist or the caller does not have WINSTATION\_QUERY permission. |

**LoginId:**  The ID of the session for which to retrieve the WinStation name.

**pWinStationName:**  Pointer to a buffer holding the session name. The length of the buffer MUST be equal to or greater than (WINSTATIONNAME\_LENGTH + 1).

**NameSize:** The size, in bytes, of the buffer where the WinStation name will be stored.

**Return Values:**  Returns TRUE if the call succeeded, or FALSE if the method failed. On failure, *pResult* indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationConnect (Opnum 10)

The **RpcWinStationConnect** method connects a user's [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) client from a given terminal server session to a different terminal server session. If there is a user connected to the client session, it will be disconnected at the end of this call. If the method succeeds, the state of the session is State\_Active as defined in the **WINSTATIONSTATECLASS** enumeration (section [2.2.1.9](#Section_b619d8aa4e8a4716954ea4d596ed37f8)).

The client indicated by *ConnectLogonId* MUST have WINSTATION\_DISCONNECT permission. Similarly, *TargetLogonId* MUST have WINSTATION\_CONNECT and WINSTATION\_DISCONNECT permissions. For each of the aforementioned permissions, the method checks whether the caller has the permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting the Access Request mask to the specific permission, and fails if the caller does not have the permission.[<178>](#Appendix_A_178" \o "Product behavior note 178)

1. BOOLEAN RpcWinStationConnect(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in] DWORD ClientLogonId,
5. [in] DWORD ConnectLogonId,
6. [in] DWORD TargetLogonId,
7. [in, size\_is(PasswordSize)] PWCHAR pPassword,
8. [in, range(0, 1024)] DWORD PasswordSize,
9. [in] BOOLEAN Wait
10. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The *hServer* argument MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:** If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000); otherwise, it MUST be an implementation-specific negative value.

**ClientLogonId:**  The session ID of the caller of this method.

**ConnectLogonId:** The ID of the session from which the connection is being made. This MUST be the same as *ClientLogonId* and MUST be an existing session ID. The user MUST be logged on. To indicate the current session, this MUST be LOGONID\_CURRENT.

**TargetLogonId:**  The session ID of the session to which the connection is being made. Cannot be the same as *ConnectLogonId* and MUST be an existing session ID.

**pPassword:**  The password of *TargetLogonId*'s session. The password MUST be valid. The password MAY be NULL if the same user is making the call as the user logged on to *TargetLogonId*'s session.

**PasswordSize:** The length of the string *pPassword* in characters including the terminating NULL character.

**Wait:**  TRUE indicates to wait for the connection to complete, FALSE otherwise.[<179>](#Appendix_A_179" \o "Product behavior note 179)

**Return Values:**  Returns TRUE if the call succeeded, or FALSE if the method failed. On failure, *pResult* indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationDisconnect (Opnum 13)

The RpcWinStationDisconnect method disconnects, on the server, the terminal server client from a session. If the method succeeds, the state of the session is State\_Disconnected as defined in the **WINSTATIONSTATECLASS** enumeration (section [2.2.1.9](#Section_b619d8aa4e8a4716954ea4d596ed37f8)).

The caller of this method MUST have WINSTATION\_DISCONNECT permission on the session to disconnect. The method checks whether the caller has WINSTATION\_DISCONNECT permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.[<180>](#Appendix_A_180" \o "Product behavior note 180)

1. BOOLEAN RpcWinStationDisconnect(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in] DWORD LogonId,
5. [in] BOOLEAN bWait
6. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). hServer MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:**  Failure error code if the call to RpcWinStationDisconnect failed. If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | The call succeeded. |
| STATUS\_ACCESS\_DENIED  0xC0000022 | The caller does not have permission to disconnect the session. |

**LogonId:**  The ID of the session to disconnect. Can be LOGONID\_CURRENT to indicate the current session.

**bWait:**  TRUE to wait for the disconnect to complete before returning, FALSE otherwise.[<181>](#Appendix_A_181" \o "Product behavior note 181)

**Return Values:**  Returns TRUE if the call succeeded, or FALSE if the method failed. On failure, *pResult* indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationReset (Opnum 14)

The RpcWinStationReset method resets a session. Resetting a session will lead to the user being logged off and his or her terminal server client being disconnected. The caller MUST have WINSTATION\_RESET permissions. The method checks whether the caller has WINSTATION\_RESET permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.[<182>](#Appendix_A_182" \o "Product behavior note 182)

1. BOOLEAN RpcWinStationReset(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in] DWORD LogonId,
5. [in] BOOLEAN bWait
6. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). hServer MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:**  Failure error code if the call to RpcWinStationReset failed. If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | The call succeeded. |
| STATUS\_ACCESS\_DENIED  0xC0000022 | The caller does not have WINSTATION\_RESET permission. |

**LogonId:**  The ID of the session to reset.

**bWait:** TRUE to wait for the disconnect to complete before returning, FALSE otherwise.

**Return Values:**  Returns TRUE if the call succeeded, or FALSE if the method failed. On failure, *pResult* indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationShutdownSystem (Opnum 15)

The RpcWinStationShutdownSystem method shuts down the system and optionally logs off all sessions and/or reboots the system. The caller requires SeShutdownPrivilege (see [[MSDN-PRVLGECNSTS]](https://go.microsoft.com/fwlink/?LinkId=212627)) when performing the shutdown locally and SeRemoteShutdownPrivilege (see [MSDN-PRVLGECNSTS]) when performing the shutdown remotely. The caller calls ExitWindowsEx (see [[MSDN-ExitWindowsEx]](https://go.microsoft.com/fwlink/?LinkId=212628)) to perform the actual shutdown once all checks have been completed.

1. BOOLEAN RpcWinStationShutdownSystem(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in] DWORD ClientLogonId,
5. [in] DWORD ShutdownFlags
6. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The *hServer* MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:**  Failure error code if the call to RpcWinStationShutdownSystem failed. If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | The call succeeded. |
| STATUS\_ACCESS\_DENIED  0xC0000022 | The caller does not have permission to shut down the system. |

**ClientLogonId:** The session requesting to shut down the system. Ignored when the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) call is remote.

**ShutdownFlags:**  Shutdown flags. It MUST be any bitwise OR combination of the following flags.

| Value | Meaning |
| --- | --- |
| WSD\_LOGOFF  0x00000001 | Forces sessions to logoff. |
| WSD\_SHUTDOWN  0x00000002 | Shuts down the system. |
| WSD\_REBOOT  0x00000004 | Reboots after shutdown. |
| WSD\_POWEROFF  0x00000008 | Powers off after shutdown. |

**Return Values:**  Returns TRUE if the call succeeded, or FALSE if the method failed. On failure, *pResult* indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationWaitSystemEvent (Opnum 16)

The RpcWinStationWaitSystemEvent method waits synchronously for a system event from an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) API request on behalf of the caller. There is no time-out on the wait. Only one event wait at a time can be posted per server handle. If an event wait is already outstanding and the new request is not a cancel, the new request will fail. The caller is not required to have any specific permission to call RpcWinStationWaitSystemEvent. The first time this is called, the server will create an event block for the handle specified by hServer. This event block will be cleared if RpcWinStationWaitSystemEvent is called with *EventMask* equal to WEVENT\_NONE or if [RpcWinStationCloseServer](#Section_89048acf28c64d469742eef51f507f05) or [RpcWinStationCloseServerEx](#Section_9d198d98b7f1475d8c29cc188f5bc0fc) are closed for the [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) *hServer*.

1. BOOLEAN RpcWinStationWaitSystemEvent(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in] DWORD EventMask,
5. [out] DWORD\* pEventFlags
6. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). hServer MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:** If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000); otherwise, it MUST be an implementation-specific negative value.

**EventMask:**  The mask of events for which to wait. It MUST be any bitwise OR combination of the following except for WEVENT\_NONE.

| Value | Meaning |
| --- | --- |
| WEVENT\_NONE  0x00000000 | The client requests to clear its event wait block. This MUST be called when completing waiting for the event. When RpcWinStationCloseServer is called for hServer, this method and mask value is called on the client's behalf. |
| WEVENT\_CREATE  0x00000001 | Wait for a new session to be created. |
| WEVENT\_DELETE  0x00000002 | Wait for an existing session to be deleted. |
| WEVENT\_RENAME  0x00000004 | Wait for a session to be renamed. |
| WEVENT\_CONNECT  0x00000008 | The session connected to a client. |
| WEVENT\_DISCONNECT  0x00000010 | A client disconnected from the session. |
| WEVENT\_LOGON  0x00000020 | A user logged on to the session. |
| WEVENT\_LOGOFF  0x00000040 | A user logged off from the session. |
| WEVENT\_STATECHANGE  0x00000080 | The session state changed. |
| WEVENT\_LICENSE  0x00000100 | The license state changed.[<183>](#Appendix_A_183" \o "Product behavior note 183) |
| WEVENT\_ALL  0x7fffffff | Wait for all event types. |
| WEVENT\_FLUSH  0x80000000 | Release all waiting clients. |

**pEventFlags:**  Pointer to a variable to receive a bitmask that is a subset of *EventMask* indicating which events actually occurred during this wait operation.

**Return Values:**  Returns TRUE if the call succeeded, or FALSE if the method failed. On failure, *pResult* indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationShadow (Opnum 17)

The RpcWinStationShadow method starts a shadow (remote control) operation of another [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) session. If the method succeeds, the state of the session that started the shadow operation is State\_Shadow and the state of the session being shadowed is State\_Active as defined in the **WINSTATIONSTATECLASS** enumeration (section [2.2.1.9](#Section_b619d8aa4e8a4716954ea4d596ed37f8)).

The caller MUST have WINSTATION\_SHADOW permission. The other session can be local or on a terminal server. The method MUST be called from inside a remote terminal server session. The session to shadow MUST be in the active state with a user logged on. The method checks whether the caller has WINSTATION\_SHADOW permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.[<184>](#Appendix_A_184" \o "Product behavior note 184)

1. BOOLEAN RpcWinStationShadow(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in] DWORD LogonId,
5. [in, unique, size\_is(NameSize)]
6. PWCHAR pTargetServerName,
7. [in, range(0, 1024)] DWORD NameSize,
8. [in] DWORD TargetLogonId,
9. [in] BYTE HotKeyVk,
10. [in] USHORT HotkeyModifiers
11. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The *hServer* argument value MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:** If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000); otherwise, it MUST be an implementation-specific negative value.

**LogonId:** The ID of the session to shadow from.

**pTargetServerName:** The shadow target server name. Set *pTargetServerName* to NULL to indicate the current server.

**NameSize:** The size of the string *pTargetServerName*, in bytes. MAY be 0 if *pTargetServerName* is NULL.

**TargetLogonId:** The shadow target session ID.

**HotKeyVk:** The virtual key code of the key to press to stop shadowing. This key is used in combination with the *HotkeyModifiers* parameter.

**HotkeyModifiers:** The virtual modifier that signifies the modifier key, such as shift or control, to press to stop shadowing. The modifier key is used in combination with the key signified by the *HotKeyVk* parameter. This parameter MAY be any combination of KBDSHIFT, KBDCTRL, and KBDALT to indicate the SHIFT key, the CTRL key, and the ALT key, respectively.

**Return Values:** Returns TRUE if the call succeeded, or FALSE if the method failed. On failure, pResult indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationBreakPoint (Opnum 29)

The RpcWinStationBreakPoint method breaks into the debugger in either the session process of a specific session or in the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) service process. When this method is called, the server impersonates the caller and then tries to enable SeShutdownPrivilege (see [[MSDN-PRVLGECNSTS]](https://go.microsoft.com/fwlink/?LinkId=212627)). If the attempt to enable this privilege fails, the RpcWinStationBreakPoint call fails.[<185>](#Appendix_A_185" \o "Product behavior note 185)

1. BOOLEAN RpcWinStationBreakPoint(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in] DWORD LogonId,
5. [in] BOOLEAN KernelFlag
6. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The *hServer* argument MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:**  Failure error code if the call to RpcWinStationBreakPoint failed. If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | The call succeeded. |
| STATUS\_ACCESS\_DENIED  0xC0000022 | The SeShutdownPrivilege (see [MSDN-PRVLGECNSTS]) privilege cannot be enabled. |

**LogonId:**  The ID of the session to break into the debugger. If this parameter is -2, the terminal server service MUST break into the debugger instead.[<186>](#Appendix_A_186" \o "Product behavior note 186)

**KernelFlag:**  Set to TRUE to indicate that the server will break into the debugger in a particular session in kernel mode. If *LogonId* is -2, the server MUST break into the debugger in user mode.

**Return Values:**  Returns TRUE if the call succeeded, or FALSE if the method failed. On failure, *pResult* indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationReadRegistry (Opnum 30)

The RpcWinStationReadRegistry method tells the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) to reread, from the registry, the configuration data for all the [**WinStations**](#gt_8cc89bb9-be94-4672-8a5d-9947b91bc952), and to update the memory locations where this data was stored with the values read from the registry.[<187>](#Appendix_A_187" \o "Product behavior note 187) The caller of this [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) method MUST be running either as [**SYSTEM**](#gt_3f2baf2c-3e0b-4764-b7a5-54cebabeb85a) or as an Administrator.[<188>](#Appendix_A_188" \o "Product behavior note 188)

1. BOOLEAN RpcWinStationReadRegistry(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult
4. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). hServer MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:**  If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000), as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90); otherwise, it MUST be an implementation-specific negative value.

**Return Values:**  Returns TRUE if the call succeeded, or FALSE if the method failed. On failure, *pResult* indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### OldRpcWinStationEnumerateProcesses (Opnum 34)

The OldRpcWinStationEnumerateProcesses method calls the [RpcWinStationEnumerateProcesses](#Section_3559bc198ca34bcf842ecf25f592dc1a) method and returns whatever is returned by that method. It has the same parameters as the RpcWinStationEnumerateProcesses method. No special permissions are required to call this method.[<189>](#Appendix_A_189" \o "Product behavior note 189)

1. BOOLEAN OldRpcWinStationEnumerateProcesses(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [out, size\_is(ByteCount)] PBYTE pProcessBuffer,
5. [in, range(0, 0x8000 )] DWORD ByteCount
6. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). hServer MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:**  Failure error code if the call to RpcWinStationEnumerateProcesses failed. If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000), as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | The call succeeded. |
| STATUS\_INFO\_LENGTH\_MISMATCH  0xC0000004 | *ByteCount* is too small to receive the data. |

**pProcessBuffer:**  Pointer to a buffer receiving the list of processes.

In *pProcessBuffer* for each process, the server will return the following in the buffer, in this order:

1. A [TS\_PROCESS\_INFORMATION\_NT4](#Section_0a8a73d93c9e4d2d91e47fef48b6c29d) structure.
2. A number of SYSTEM\_THREAD\_INFORMATION (see [WININTERNALS] or [[MSFT-WINSYSINTERNALS]](https://go.microsoft.com/fwlink/?LinkId=208768)) structures equal to the maximum number of threads in the process.
3. A [TS\_SYS\_PROCESS\_INFORMATION](#Section_96702c7d6f5f49658f295fdbea73278e) structure for the process.

Only up to SIZEOF TS4\_SYSTEM\_THREAD\_INFORMATION of the SYSTEM\_THREAD\_INFORMATION structure (as specified in (see [WININTERNALS] or [MSFT-WINSYSINTERNALS]) is copied into the buffer and only up to SIZEOF TS4\_SYSTEM\_PROCESS\_INFORMATION of the TS\_SYS\_PROCESS\_INFORMATION\_NT4 structure is copied into the buffer.

**ByteCount:**  The size, in bytes, of the *pProcessBuffer* parameter. If ByteCount is too small to receive the data, the method returns an error code (STATUS\_INFO\_LENGTH\_MISMATCH, as specified in [MS-ERREF]) in the *pResult* parameter. Note that during failure no indication is given to the caller specifying the correct size if *pProcessBuffer* is too small.

**Return Values:**  Returns TRUE if the call succeeded, or FALSE if the method failed. On failure, *pResult* indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationEnumerateProcesses (Opnum 36)

The RpcWinStationEnumerateProcesses method retrieves the processes running on a remote server on which the caller has WINSTATION\_QUERY permission to retrieve information. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.[<190>](#Appendix_A_190" \o "Product behavior note 190)

1. BOOLEAN RpcWinStationEnumerateProcesses(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [out, size\_is(ByteCount)] PBYTE pProcessBuffer,
5. [in, range(0, 0x8000 )] DWORD ByteCount
6. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). hServer MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:**  Failure error code if the call to RpcWinStationEnumerateProcesses failed. If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000), as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | The call succeeded. |
| STATUS\_INFO\_LENGTH\_MISMATCH  0xC0000004 | *ByteCount* is too small to receive the data. |

**pProcessBuffer:**  Pointer to a buffer receiving the list of processes.

In *pProcessBuffer* for each process, the server will return the following in the buffer, in this order:

1. A [TS\_PROCESS\_INFORMATION\_NT4](#Section_0a8a73d93c9e4d2d91e47fef48b6c29d) structure.
2. A number of SYSTEM\_THREAD\_INFORMATION (see [WININTERNALS] or [[MSFT-WINSYSINTERNALS]](https://go.microsoft.com/fwlink/?LinkId=208768)) structures equal to the maximum number of threads in the process.
3. A [TS\_SYS\_PROCESS\_INFORMATION](#Section_96702c7d6f5f49658f295fdbea73278e) structure for the process.

Only up to SIZEOF TS4\_SYSTEM\_THREAD\_INFORMATION of the SYSTEM\_THREAD\_INFORMATION structure (as specified in [WININTERNALS] or [MSFT-WINSYSINTERNALS]) is copied into the buffer and only up to SIZEOF TS4\_SYSTEM\_PROCESS\_INFORMATION of the TS\_SYS\_PROCESS\_INFORMATION\_NT4 structure is copied into the buffer.

**ByteCount:**  The size, in bytes, of the *pProcessBuffer* parameter. If ByteCount is too small to receive the data, the method returns an error code (STATUS\_INFO\_LENGTH\_MISMATCH, as specified in [MS-ERREF]) in the *pResult* parameter. Note that during failure no indication is given to the caller specifying the correct size if *pProcessBuffer* is too small.

**Return Values:**  Returns TRUE if the call succeeded, or FALSE if the method failed. On failure, *pResult* indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationTerminateProcess (Opnum 37)

The RpcWinStationTerminateProcess method terminates the specified process. An attempt is made to enable the SE\_DEBUG\_PRIVILEGE privilege to kill processes not owned by the current user, including processes running in other [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) sessions. Caller MUST have terminate permission to terminate the process.

1. BOOLEAN RpcWinStationTerminateProcess(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in] DWORD ProcessId,
5. [in] DWORD ExitCode
6. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). hServer MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:**  Failure error code if the call to RpcWinStationTerminateProcess failed. If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000), as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | The call succeeded. |
| STATUS\_ACCESS\_DENIED  0xC0000022 | The caller does not have permission to terminate the process. |

**ProcessId:** The ID of the process to terminate.

**ExitCode:** The exit code to be used by the process and threads that are terminated as a result of this call.

**Return Values:**  Returns TRUE if the call succeeded, and FALSE if the method failed. On failure, *pResult* indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationGetAllProcesses (Opnum 43)

The RpcWinStationGetAllProcesses method retrieves the list of processes running on the server machine. Only the processes from the sessions on which the user has WINSTATION\_QUERY permission will be retrieved. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. BOOLEAN RpcWinStationGetAllProcesses(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in] ULONG Level,
5. [in, out] BOUNDED\_ULONG\* pNumberOfProcesses,
6. [out, size\_is(,\*pNumberOfProcesses)]
7. PTS\_ALL\_PROCESSES\_INFO\* ppTsAllProcessesInfo
8. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The *hServer* argument MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:**  Failure error code if the call to RpcWinStationGetAllProcesses failed. If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | The call was successful. |
| STATUS\_ACCESS\_DENIED  0xC0000022 | The caller does not have permission. |

**Level:**  MUST be 0. Any other value will lead to FALSE being returned by the method.

**pNumberOfProcesses:**  The number of processes requested by the caller. On return, this indicates the number of processes actually stored in the *ppTsAllProcessesInfo* parameter.

**ppTsAllProcessesInfo:**  Pointer to an array of processes allocated and returned by the method. \***ppTsAllProcessesInfo** is allocated by the method to be an array of [TS\_ALL\_PROCESSES\_INFO](#Section_8f5564a61b5e4381afc44acf7ad39d6d) structures. The array returned by the method MUST be freed by the caller.

**Return Values:**  Returns TRUE if the call succeeded, and FALSE if the lookup failed. On failure, pResult indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationGetProcessSid (Opnum 44)

The RpcWinStationGetProcessSid method retrieves the process security identifier (SID), as specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.2, for a given process ID and process start time combination.[<191>](#Appendix_A_191" \o "Product behavior note 191)The caller MUST have the PROCESS\_QUERY\_INFORMATION access right to the process being queried and the TOKEN\_QUERY access right to the access token associated with the process. For more information on the process access rights, see [[MSDN-PROCRIGHTS]](https://go.microsoft.com/fwlink/?LinkId=208137). For more information on access rights for access tokens, see [[MSDN-TOKENRIGHTS]](https://go.microsoft.com/fwlink/?LinkId=208138).

1. BOOLEAN RpcWinStationGetProcessSid(
2. [in] SERVER\_HANDLE hServer,
3. [in] DWORD dwUniqueProcessId,
4. [in] LARGE\_INTEGER ProcessStartTime,
5. [out] LONG\* pResult,
6. [in, out, unique, size\_is(dwSidSize)]
7. PBYTE pProcessUserSid,
8. [in, range(0, 1024 )] DWORD dwSidSize,
9. [in, out] DWORD\* pdwSizeNeeded
10. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The *hServer* argument MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**dwUniqueProcessId:**  The process ID to retrieve the SID.

**ProcessStartTime:**  The start time of the process indicated by *dwUniqueProcessId*. This is a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601 (UTC). *ProcessStartTime* combined with *dwUniqueProcessId* is used to identify a process.

**pResult:**  Failure error code if the call to RpcWinStationGetProcessSid failed. If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | The call succeeded. |
| 0x8007007A | The size of *pProcessUserSid* buffer is too small.[<192>](#Appendix_A_192" \o "Product behavior note 192) |
| STATUS\_BUFFER\_TOO\_SMALL  0xC0000023 | The size of pProcessUserSid buffer is too small.[<193>](#Appendix_A_193" \o "Product behavior note 193) |
| 0x80070005 | The caller does not have necessary permissions.[<194>](#Appendix_A_194" \o "Product behavior note 194) |
| STATUS\_ACCESS\_DENIED  0xC0000022 | The caller does not have necessary permissions.[<195>](#Appendix_A_195" \o "Product behavior note 195) |

**pProcessUserSid:**  The buffer into which the method MUST copy the SID of the process. MUST be NULL if **dwSidSize** is zero, in which case the correct size will be returned in **pdwSizeNeeded**.

**dwSidSize:**  The size of the buffer, in bytes, pointed to by **pProcessUserSid**. If the buffer is too small, STATUS\_BUFFER\_TOO\_SMALL[<196>](#Appendix_A_196" \o "Product behavior note 196) or 0x8007007A[<197>](#Appendix_A_197" \o "Product behavior note 197) is returned in **pResult**.

**pdwSizeNeeded:**  Indicates the length of the SID. If STATUS\_BUFFER\_TOO\_SMALL[<198>](#Appendix_A_198" \o "Product behavior note 198) or 0x8007007A[<199>](#Appendix_A_199" \o "Product behavior note 199) is returned in **pResult**, this indicates to the caller the correct size to allocate to a buffer prior to calling the method again.

**Return Values:**  Returns TRUE if the call succeeded, and FALSE if the method failed. On failure, **pResult** indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationGetTermSrvCountersValue (Opnum 45)

The RpcWinStationGetTermSrvCountersValue method retrieves the current value of requested [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) performance counters. The caller is not required to have any specific permission to call this method.

1. BOOLEAN RpcWinStationGetTermSrvCountersValue(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in, range(0, 0x1000)] DWORD dwEntries,
5. [in, out, size\_is(dwEntries)] PTS\_COUNTER pCounter
6. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The *hServer* argument MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:**  If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000); otherwise it MUST be an implementation-specific negative value.

**dwEntries:**  The number of performance counters to query. Indicates the size of the array pointed to by pCounter.

**pCounter:**  An array of [TS\_COUNTER](#Section_a0173488c4834340ab1aaab2d21ea323) structures. The caller MUST set the **dwCounterId** in the TS\_COUNTER structures for each entry in the array to indicate the counter whose current value to retrieve. On return, the method MUST set the value for that performance counter. If the performance counter ID is not recognized or is not supported, it will set the **bResult** to 0.

**Return Values:**  Returns TRUE if the call succeeded, and FALSE if the method failed. On failure, pResult indicates the failure status code. Individual entries in the array pCounter will indicate whether or not the counter data for that counter could be retrieved.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationReInitializeSecurity (Opnum 46)

The RpcWinStationReInitializeSecurity method reinitializes security for all non-console [**WinStations**](#gt_8cc89bb9-be94-4672-8a5d-9947b91bc952) (remote connection protocols). [<200>](#Appendix_A_200" \o "Product behavior note 200) Existing sessions will not be affected, but future sessions will have the new security descriptor read from the registry applied to them. This method MUST be called by processes running as SYSTEM or as an Administrator.[<201>](#Appendix_A_201" \o "Product behavior note 201)

1. BOOLEAN RpcWinStationReInitializeSecurity(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult
4. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The *hServer* argument MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:**  If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000); otherwise, it MUST be an implementation-specific negative value.

**Return Values:**  Returns TRUE if the call succeeded, and FALSE if the method failed. On failure, pResult indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationGetLanAdapterName (Opnum 53)

The RpcWinStationGetLanAdapterName method returns the name of the LAN adapter with a specific LAN adapter number (lana) and transport type, if it is configured to be used for a Terminal Services protocol connection. No special permissions are required to call this method.[<202>](#Appendix_A_202" \o "Product behavior note 202)[<203>](#Appendix_A_203" \o "Product behavior note 203)

1. BOOLEAN RpcWinStationGetLanAdapterName(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in, range(0, 0x1000)] DWORD PdNameSize,
5. [in, size\_is(PdNameSize)] PWCHAR pPdName,
6. [in, range(0, 1024)] ULONG LanAdapter,
7. [out] ULONG\* pLength,
8. [out, size\_is(,\*pLength)] PWCHAR\* ppLanAdapter
9. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The *hServer* argument MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:** If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000); otherwise, it MUST be an implementation-specific negative value.

**PdNameSize:** The size, in bytes, of *pPdName* including the terminating NULL character.

**pPdName:** The transport protocol type on which to retrieve information. This MUST be any of the following strings: tcp, netbios, ipx, spx.

**LanAdapter:** The number of the LAN adapter to retrieve information (also known as lana). If this is set to "0", it will always return a LAN adapter name to indicate all LAN adapters configured with the protocol, irrespective of the transport protocol type specified in *pPdName*.

**pLength:** The pointer to a ULONG containing the length of the string *ppLanAdapter*, in characters including the terminating NULL character. If *LanAdapter* is 0, this value MUST be DEVICENAME\_LENGTH+1.

**ppLanAdapter:**  The pointer to a string allocated by this method for retrieving the LAN adapter's name. This memory MUST be freed by the caller.

**Return Values:**  Returns TRUE if the call is successful, and FALSE if the method fails. On failure, pResult indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationUpdateSettings (Opnum 58)

The RpcWinStationUpdateSettings method rereads settings for all [**WinStations**](#gt_8cc89bb9-be94-4672-8a5d-9947b91bc952). The caller MUST have WINSTATION\_QUERY permission. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.[<204>](#Appendix_A_204" \o "Product behavior note 204)

1. BOOLEAN RpcWinStationUpdateSettings(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in] DWORD SettingsClass,
5. [in] DWORD SettingsParameters
6. );

**hServer:**  A handle to the server object of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The *hServer* argument MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:**  Failure error code if the call to RpcWinStationUpdateSettings failed. If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | The call was successful. |
| STATUS\_ACCESS\_DENIED  0xC0000022 | The caller does not have permission to read the settings. |
| STATUS\_INVALID\_PARAMETER  0xC000000D | Unrecognized *SettingsClass*. |

**SettingsClass:**  The class for which to update settings.

| Value | Meaning |
| --- | --- |
| WINSTACFG\_SESSDIR  0x00000001 | Contacts Session Directory to reread the WinStation settings.[<205>](#Appendix_A_205" \o "Product behavior note 205) |
| WINSTACFG\_LEGACY  0x00000000 | Rereads settings from the local registry for the configured winstations. This does the same thing as [RpcWinStationReadRegistry](#Section_5d74a6b079fd4091a2eb4974131b4ea4). |

**SettingsParameters:** MUST be 0.

**Return Values:** Returns TRUE if the call succeeded, and FALSE if the method failed. On failure, pResult indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationShadowStop (Opnum 59)

The RpcWinStationShadowStop method stops all shadow operations on the specified session, including whether the session is acting as a shadow client (a session that is shadowing another session) or as a shadow target (a session being shadowed by another session). If the method succeeds, both the state of the session that started the shadow operation and the state of the session being shadowed are State\_Active as defined in the **WINSTATIONSTATECLASS** enumeration (section [2.2.1.9](#Section_b619d8aa4e8a4716954ea4d596ed37f8)).

Caller MUST have WINSTATION\_DISCONNECT and WINSTATION\_RESET permissions. For each aforementioned required permission, the method checks whether the caller has the permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting the Access Request mask to the specific permission, and fails if the caller does not have the permission.[<206>](#Appendix_A_206" \o "Product behavior note 206)

1. BOOLEAN RpcWinStationShadowStop(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in] DWORD LogonId,
5. [in] BOOLEAN bWait
6. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The hServer argument MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:** Failure error code if the call to RpcWinStationShadowStop failed. If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | The call was successful. |
| STATUS\_CTX\_WINSTATION\_NOT\_FOUND  0xC00A0015 | Indicates the session does not exist. |
| STATUS\_CTX\_SHADOW\_NOT\_RUNNING  0xC00A0036 | Indicates the session is either not active or not being shadowed. |
| STATUS\_ACCESS\_DENIED  0xC0000022 | Indicates the caller does not have permission to end shadowing on the session. |

**LogonId:** The ID of the session on which to stop shadowing operations.

**bWait:** TRUE indicates wait for reset to complete, FALSE otherwise.

**Return Values:** Returns TRUE if the call succeeded, and FALSE if the method failed. On failure, *pResult* indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationCloseServerEx (Opnum 60)

The RpcWinStationCloseServerEx method closes the server [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee) for [**WinStation**](#gt_8cc89bb9-be94-4672-8a5d-9947b91bc952) APIs. The call to this method MUST be serialized if there are multiple threads running; otherwise, the behavior of this function is unknown. No special permissions are required to call this method.

1. BOOLEAN RpcWinStationCloseServerEx(
2. [in, out] SERVER\_HANDLE\* phServer,
3. [out] DWORD\* pResult
4. );

**phServer:** Pointer to a variable that is a handle to the server. The variable is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The handle MUST be returned from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a). On return from this method, \*phServer is set to NULL.

**pResult:** If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000); otherwise, it MUST be an implementation-specific negative value.

**Return Values:** Returns TRUE if the call succeeded, and FALSE if the call failed. On failure, pResult indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationIsHelpAssistantSession (Opnum 61)

The RpcWinStationIsHelpAssistantSession method determines if a session is created by the built-in HelpAssistant user account.[<207>](#Appendix_A_207" \o "Product behavior note 207) The caller is not required to have any specific permission to call this method.

1. BOOLEAN RpcWinStationIsHelpAssistantSession(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in] ULONG SessionId
5. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The *hServer* argument MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:**  Failure error code if the call to RpcWinStationIsHelpAssistantSession failed. If no error was encountered while executing the call, this parameter MUST be STATUS\_SUCCESS (0x00000000).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | The call was executed successfully. |
| STATUS\_UNSUCCESSFUL  0xC0000001 | Execution of the call failed. |
| STATUS\_CTX\_WINSTATION\_NOT\_FOUND  0xC00A0015 | The *SessionId* does not exist. |
| STATUS\_WRONG\_PASSWORD  0xC000006A | This is a Help Assistant session but the help assistance ticket associated with the session is no longer valid. |

**SessionId:** The ID of the session to check.

**Return Values:**  Returns TRUE if the session is running as HelpAssistant, and FALSE if this is not a HelpAssistant session or if an error was encountered during the test. On failure, pResult indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationGetMachinePolicy (Opnum 62)

The RpcWinStationGetMachinePolicy method returns a copy of the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) machine policy to the caller.[<208>](#Appendix_A_208" \o "Product behavior note 208) The caller is not required to have any specific permission to call this method.

1. BOOLEAN RpcWinStationGetMachinePolicy(
2. [in] SERVER\_HANDLE hServer,
3. [in, out, size\_is(bufferSize)] PBYTE pPolicy,
4. [in, range(0, 0x8000 )] ULONG bufferSize
5. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The *hServer* argument MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pPolicy:**  Pointer to a buffer to receive the machine policy. This buffer MUST be of type [POLICY\_TS\_MACHINE](#Section_d7cb8c2df60e42d6b8f4e617ad4d8c1b).

**bufferSize:**  Size of the buffer, in bytes, pointed to by *pPolicy*. This size MUST NOT be less than sizeof(POLICY\_TS\_MACHINE).

**Return Values:**  Returns TRUE if the call succeeded, and FALSE if the method failed.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationCheckLoopBack (Opnum 65)

The RpcWinStationCheckLoopBack method checks if there is a loopback when a client tries to connect. Loopback refers to opening a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) session on the local machine.[<209>](#Appendix_A_209" \o "Product behavior note 209) The caller is not required to have any specific permission to call this method.

1. BOOLEAN RpcWinStationCheckLoopBack(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in] DWORD ClientLogonId,
5. [in] DWORD TargetLogonId,
6. [in, size\_is(NameSize)] PWCHAR pTargetServerName,
7. [in, range(0, 1024)] DWORD NameSize
8. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The *hServer* argument MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:** Failure error code if the call to RpcWinStationCheckLoopBack failed. If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | The call was successful. |
| STATUS\_ACCESS\_DENIED  0xC0000022 | A loopback was detected. |
| STATUS\_CTX\_WINSTATION\_ACCESS\_DENIED  0xC00A002B | The server is in the process of shutting down and cannot complete the request. |

**ClientLogonId:** The ID of the session from which the terminal server client was started.

**TargetLogonId:** The session ID to which the client is trying to connect.

**pTargetServerName:** The name of the target server to which the client is connecting. The string MUST contain the terminating NULL character.

**NameSize:** The length of the *pTargetServerName* string in characters including the terminating NULL character.

**Return Values:**  Returns FALSE if there is no loopback, and TRUE if a loopback was detected. This method returns TRUE also in the case when an error was encountered during the loopback test. In this case, the *pResult* value contains the relevant error code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | A loopback was detected or the method call failed. |
| 0x00  FALSE | A loopback was not detected. |

##### RpcConnectCallback (Opnum 66)

The RpcConnectCallback method initiates a TCP connection to the specified IP address and waits for the party on the other end of the connection to start the [**Remote Desktop Protocol (RDP)**](#gt_17c795a6-68bf-46bf-8ea8-467c8df1a0b3) connection sequence. More information on Remote Desktop Protocol can be found in [[MS-RDPBCGR]](%5bMS-RDPBCGR%5d.pdf#Section_5073f4ed1e9345e1b0396e30c385867c). This method MUST be called by processes running as SYSTEM. Note that this function assumes that the address being passed in is an IPv4 address. IPv6 addresses are not supported.[<210>](#Appendix_A_210" \o "Product behavior note 210)

1. BOOLEAN RpcConnectCallback(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in] DWORD TimeOut,
5. [in] ULONG AddressType,
6. [in, size\_is(AddressSize)] PBYTE pAddress,
7. [in, range(0, 0x1000 )] ULONG AddressSize
8. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The *hServer* argument MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:**  Failure error code if the call to RpcConnectCallback failed. If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | The call was successful. |
| STATUS\_NOT\_SUPPORTED  0xC00000BB | *AddressType* is not TDI\_ADDRESS\_TYPE\_IP. This is a standard representation of a type for an IP address. For more information, see [[MSDN-TDIADDRESS]](https://go.microsoft.com/fwlink/?LinkId=93316). |
| STATUS\_INVALID\_PARAMETER  0xC000000D | *AddressSize* is not TDI\_ADDRESS\_LENGTH\_IP. For more information, see [MSDN-TDIADDRESS]. |
| STATUS\_ACCESS\_DENIED  0xC0000022 | The caller is not SYSTEM. |

**TimeOut:** Not used.

**AddressType:**  MUST be TDI\_ADDRESS\_TYPE\_IP. For more information, see [MSDN-TDIADDRESS].

**pAddress:**  Pointer to the address itself. MUST be TDI\_ADDRESS\_IP. This is a standard representation for an IP address. For more information, see [MSDN-TDIADDRESS].

**AddressSize:**  MUST be TDI\_ADDRESS\_LENGTH\_IP. This is a standard representation for the length of an IP address. For more information, see [MSDN-TDIADDRESS].

**Return Values:**  Returns TRUE if the call succeeded, and FALSE if the method failed. On failure, pResult indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationGetAllProcesses\_NT6 (Opnum 70)

The RpcWinStationGetAllProcesses\_NT6 method retrieves the processes running a remote server machine. Only the processes from the sessions on which the caller has WINSTATION\_QUERY permission will be retrieved. The method checks whether the caller has WINSTATION\_QUERY permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) by setting it as the Access Request mask, and fails if the caller does not have the permission.

1. BOOLEAN RpcWinStationGetAllProcesses\_NT6(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in] ULONG Level,
5. [in, out] BOUNDED\_ULONG\* pNumberOfProcesses,
6. [out, size\_is(,\*pNumberOfProcesses)]
7. PTS\_ALL\_PROCESSES\_INFO\_NT6\* ppTsAllProcessesInfo
8. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The *hServer* argument MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:**  Failure error code if the call to RpcWinStationGetAllProcesses\_NT6 failed. If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | The call was successful. |
| STATUS\_ACCESS\_DENIED  0xC0000022 | The caller does not have permission for the operation. |

**Level:**  MUST be GAP\_LEVEL\_BASIC (0). Any other value will lead to the method returning FALSE.

**pNumberOfProcesses:**  The number of processes requested by the caller. On return, this indicates the number of processes actually stored in the ppTsAllProcessesInfo parameter.

**ppTsAllProcessesInfo:**  Pointer to an array of processes allocated and returned by the method. \*ppTsAllProcessesInfo is allocated by the method to be an array of [TS\_ALL\_PROCESSES\_INFO\_NT6](#Section_d60fc5882cc744649c53879142f5a58d) structures. The array returned by the method MUST be freed by the caller.

**Return Values:**  Returns TRUE if the call succeeded, and FALSE if the lookup failed. On failure, pResult indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

##### RpcWinStationOpenSessionDirectory (Opnum 75)

The RpcWinStationOpenSessionDirectory method pings the Session Directory to see if it can accept [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) calls. The caller MUST be either SYSTEM or an [**administrator**](#gt_eb335f5b-619b-46ed-9138-06e9910108c3). The method performs access checks as defined in sections [3.1.2](#Section_44ae75e5641442258d5331426d7990a8) and [3.1.3](#Section_d8999c6c1ce34e9cb24d464651a2904b). The method fails if both checks fail. For more information about the Session Directory, see [[MSFT-SDLBTS]](https://go.microsoft.com/fwlink/?LinkId=90204).[<211>](#Appendix_A_211" \o "Product behavior note 211)

1. BOOLEAN RpcWinStationOpenSessionDirectory(
2. [in] SERVER\_HANDLE hServer,
3. [out] DWORD\* pResult,
4. [in, string, max\_is(64)] PWCHAR pszServerName
5. );

**hServer:** Handle to the server object. This is of type [SERVER\_HANDLE](#Section_49484cdf2feb4de9a5885d01cf8995dc). The *hServer* argument MUST be obtained from a previous call to [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a).

**pResult:**  Failure error code if the call to RpcWinStationOpenSessionDirectory failed. If the call was successful, this parameter MUST be STATUS\_SUCCESS (0x00000000).

| Value | Meaning |
| --- | --- |
| STATUS\_SUCCESS  0x00000000 | The call was successful. |
| STATUS\_UNSUCCESSFUL  0xC0000001 | The server is not in [**application server mode**](#gt_b95e0f34-13ff-496c-8cad-bf2fe26a5b59) on an advanced servers Stock Keeping Unit (SKU). |
| STATUS\_ACCESS\_DENIED  0xC0000022 | The caller is not SYSTEM nor an administrator. |

**pszServerName:** The name of the server hosting session directory to which to attempt connection.

**Return Values:**  Returns TRUE if the call succeeded, and FALSE if the method failed. On failure, pResult indicates the failure status code.

| Return value/code | Description |
| --- | --- |
| 0x01  TRUE | Successful completion. |
| 0x00  FALSE | Method call failed. |

### Timer Events

None.

### Other Local Events

None.

## Virtual IP Client Detail

### Abstract Data Model

None.

### Timers

No protocol timers are required beyond those used internally by [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) to implement resiliency to network outages, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15).

### Initialization

The client MUST create an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) connection to the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998) by using the details specified in section [2.1](#Section_ac0535d0a8474386ae3ee93a50ff2484).

### Message Processing Events and Sequencing Rules

When a method completes, the values returned by the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) MUST be returned unmodified to the upper layer.

### Timer Events

None.

### Other Local Events

None.

## Virtual IP Server Detail

### Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

#### Abstract Data Types

**TSVIPSession**: Structure containing the IP address assigned to the session. It is described in section [2.2.2.50](#Section_92043576fad1495c9a5df90185959e5f) and is used with [RpcGetSessionIP](#Section_bf669418ceeb4df399c946bc70d3942a).

### Timers

None.

### Initialization

Parameters necessary to initialize the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) protocol are specified in section [2.1](#Section_ac0535d0a8474386ae3ee93a50ff2484).

### Message Processing Events and Sequencing Rules

This protocol asks the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) runtime to perform a strict Network Data Representation (NDR) data consistency check at target level 6.0 for all methods unless otherwise specified, as defined in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 1.3.

When a method completes, the values returned by RPC MUST be returned unmodified to the upper layer. The methods MAY throw an exception and the [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b) client MUST handle these exceptions by returning the unmodified exception code to the upper layer.

#### TSVIPPublic

The TSVIPPublic provides methods to retrieve IP assigned to the session.

For information about endpoints, [**UUID**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) values, and versions, see sections [2.1](#Section_ac0535d0a8474386ae3ee93a50ff2484) and [1.9](#Section_4f5ff56828874c508b3638bb85170b58).

Methods in RPC Opnum Order

| Method | Description |
| --- | --- |
| [RpcGetSessionIP](#Section_bf669418ceeb4df399c946bc70d3942a) | Returns IP address assigned to the session.  Opnum: 0 |

##### RpcGetSessionIP (Opnum 0)

The **RpcGetSessionIP** method retrieves the IP address assigned to the session. This MUST be called by an [**administrator**](#gt_eb335f5b-619b-46ed-9138-06e9910108c3) or the same user who logged onto the session.[<212>](#Appendix_A_212" \o "Product behavior note 212) The method performs access checks as defined in section [3.1.3](#Section_d8999c6c1ce34e9cb24d464651a2904b) and [3.1.4](#Section_2b247f47fc8a41bc8735571112317e2c). The method fails if both checks fail.

1. HRESULT RpcGetSessionIP(
2. [in] handle\_t hBinding,
3. [in] USHORT Family,
4. [in] DWORD SessionId,
5. [ref, out] TSVIPSession\* ppVIPSession
6. );

**hBinding:** The [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee). For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**Family:** MUST be AF\_INET.

**SessionId:** The identifier of the session to open. This session MUST be present on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). This MUST NOT be the session ID of any of the [**listener**](#gt_1e5aa0c2-4617-4d7a-969e-a7091c415c8d) sessions.

**ppVIPSession:** The session structure containing the IP address assigned to the session. This is of type [TSVIPSession](#Section_92043576fad1495c9a5df90185959e5f).

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

### Timer Events

None.

### Other Local Events

None.

## SessEnv Details

### Abstract Data Model

None.

### Timers

None.

### Initialization

Parameters necessary to initialize the RPC protocol are specified in section [2.1](#Section_ac0535d0a8474386ae3ee93a50ff2484).

### Message Processing Events and Sequencing Rules

This protocol asks the RPC runtime to perform a strict NDR data consistency check at target level 6.0 for all methods unless otherwise specified, as defined in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 1.3.

When a method completes, the values returned by RPC MUST be returned unmodified to the upper layer. The methods MAY throw an exception, and the Terminal Services client MUST handle these exceptions by returning the unmodified exception code to the upper layer.

#### SessEnvPublicRpc

The SessEnvPublicRpc interface provides methods to call into the SessionEnv service remotely.

For information about endpoints, UUID values, and versions, see sections [2.1](#Section_ac0535d0a8474386ae3ee93a50ff2484) and [1.9](#Section_4f5ff56828874c508b3638bb85170b58).

Methods in RPC Opnum Order

| Method | Description |
| --- | --- |
| [RpcShadow2](#Section_70cb89cf10d0429e8f8bc2c4eea5eb8c) | Creates a Windows Desktop Sharing API invitation in the specified target session.  Opnum: 0 |

##### RpcShadow2 (Opnum 0)

The **RpcShadow2** method will create a shadow session using the Windows Desktop Sharing API in the target session and return an invitation to that session.

The caller MUST have WINSTATION\_SHADOW permission. The other session can be local or on a terminal server. The session to be shadowed MUST be in the active state with a user logged on. The method checks whether the caller has WINSTATION\_SHADOW permission (section [3.1.1](#Section_4507e852be3d45b6bc478ca0b552c94d)) and fails if the caller does not have the permission.

1. HRESULT RpcShadow2(
2. [in] handle\_t hBinding,
3. [in] ULONG TargetSessionId,
4. [in] SHADOW\_CONTROL\_REQUEST eRequestControl,
5. [in] SHADOW\_PERMISSION\_REQUEST eRequestPermission,
6. [out] SHADOW\_REQUEST\_RESPONSE\* pePermission,
7. [out, string, size\_is(cchInvitation)]
8. LPWSTR pszInvitation,
9. [in, range(1,8192)] ULONG cchInvitation
10. );

**hBinding:** The RPC binding handle. For more information, see [[MSDN-RPCBIND]](https://go.microsoft.com/fwlink/?LinkId=93771).

**TargetSessionId:** The ID of the session to be shadowed.

**eRequestControl:** Specifies a request for either a UI interaction or a view-only session.

**eRequestPermission:** Specifies whether to request permission before the shadow session is started. The call is synchronous, so if permission is requested, the call will wait until the user responds to the request.

**pePermission:** User response to permission request. If the response is anything other than SHADOW\_REQUEST\_RESPONSE\_ALLOW, the shadow session has been denied.

**pszInvitation:** The output data containing the invitation string for the shadow session. The data returned is an invitation string in an XML format that can be used with the Windows Desktop Sharing API IRDPSRAPIViewer::Connect method to connect to the session running in the target session (specified by **TargetSessionId**). The caller must allocate a buffer to hold this data and specify the size of the buffer in **cchInvitation**.

**cchInvitation:** The size, in WCHARs (16-bit Unicode), of **pszInvitation**.

**Return Values:** The method MUST return S\_OK (0x00000000) on success; otherwise, it MUST return an implementation-specific negative value.

| Return value/code | Description |
| --- | --- |
| 0x00000000  S\_OK | Successful completion. |

### Timer Events

None.

### Other Local Events

None.

# Protocol Examples

## LSM Enumeration Example

The following example shows how to enumerate sessions on a [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). This example uses [TermSrvBindSecure](#Section_bc8c6536baff466bb12e435b66577bbd) from section 4.3.

1. Get the LSM Binding.
2. HANDLE GetLSMBinding(LPWSTR pszServerName)
3. {
4. HANDLE hLSMBinding = NULL;
5. RPC\_STATUS rpcStatus = RPC\_S\_OK;
6. //ASSERT( NULL != pszServerName );
7. rpcStatus = TermSrvBindSecure(
8. gpszPublicUuid,
9. gpszRemoteProtocolSequence,
10. pszServerName,
11. TSRPC\_REMOTE\_ENDPOINT,
12. gpszOptions,
13. &hLSMBinding
14. );
15. if( rpcStatus != RPC\_S\_OK || hLSMBinding == NULL)
16. {
17. wprintf(L"ERR: TermSrvBindSecure failed: %d\n",
18. rpcStatus );
19. SetLastError( rpcStatus );
20. }
21. return hLSMBinding;
22. }
23. Enumerate the sessions.
24. RpcTryExcept
25. {
26. hr = RpcOpenEnum( hLSMBind, &hEnum );
27. if(hr == S\_OK)
28. {
29. hr = RpcGetEnumResult( hEnum, &pAllSessions,
30. CURRENT\_ENUM\_LEVEL, &Entries );
31. if(hr == S\_OK)
32. {
33. for(ULONG i=0;i<Entries;i++)
34. {
35. wprintf(L"%-10d %-20s %-40s\n",
36. pAllSessions[i].Data.SessionEnum\_Level3.SessionId,
37. WinstationStateClassNames[pAllSessions[i].Data.
38. SessionEnum\_Level3.State],
39. pAllSessions[i].Data.SessionEnum\_Level3.Name);
40. if( NULL != pAllSessions[i].Data.SessionEnum\_Level3.
41. pProtocolData )
42. {
43. MIDL\_user\_free( pAllSessions[i].Data.
44. SessionEnum\_Level3.pProtocolData );
45. }
46. }
47. rv1 = TRUE;
48. }
49. else
50. {
51. wprintf(L"ERR: RpcGetEnumResult failed %d\n",hr );
52. }
53. if ( pAllSessions )
54. {
55. MIDL\_user\_free(pAllSessions);
56. }
57. }
58. else
59. {
60. wprintf(L"ERR: RpcOpenEnum failed %d\n",hr );
61. }
62. }
63. RpcExcept(I\_RpcExceptionFilter(RpcExceptionCode()))
64. {
65. wprintf(L"ERR: RPC Exception %d\n",RpcExceptionCode() );
66. }
67. RpcEndExcept
69. Close the handles.
70. if(hEnum)
71. RpcCloseEnum(&hEnum);
72. if(hLSMBind)
73. RpcBindingFree(&hLSMBind);

The following diagram illustrates the message sequence for enumerating the sessions.

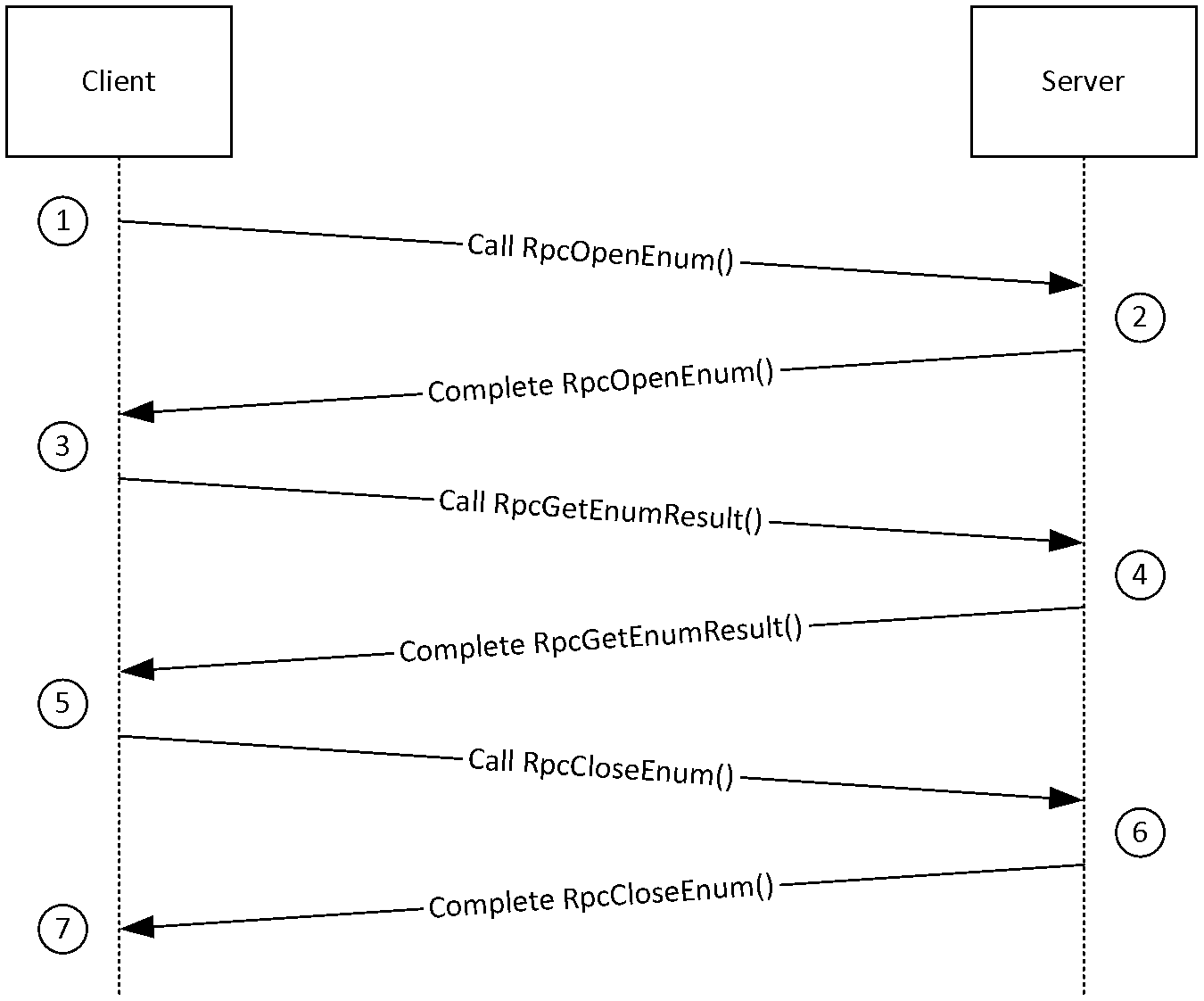


Figure 1: LSM session enumeration sequence

The sequence of messages for enumerating sessions on the server is as follows:

1. After an RPC binding has been established to the server, the client requests a session enumeration handle to be opened by the server by calling the [RpcOpenEnum](#Section_f17f4b8d8d524538acf3cdfbe581a99b) method.
2. The server, in response, opens a handle of the type [ENUM\_HANDLE](#Section_15f02d59200a435da07b055e1276d01c) and returns to the client.
3. The client then calls the [RpcGetEnumResult](#Section_1a7d5d1d1ce5448fbb4ac79741982edb) method by passing this handle, along with an uninitialized buffer, to get the list of sessions.
4. The server, on receiving the request, allocates memory for the buffer and fills it with an array of [SESSIONENUM](#Section_fd8759ef4f6c45c6b5429d5e53e90f7d) structures containing session information, one for each session on the server. It also returns the number of sessions on the server.
5. The client, on receiving the data, calls the [RpcCloseEnum](#Section_cb9386d5d99e46ff8199563f6b59ecb8) method to inform the server to close the enumeration handle.
6. The server, on receiving the RpcCloseEnum call, closes the enumeration handle.
7. The client frees the array of SESSIONENUM structures it received before exiting.

## TermService Listener Example

The following example retrieves the [**listeners**](#gt_1e5aa0c2-4617-4d7a-969e-a7091c415c8d) that run on the [**terminal server**](#gt_b416f72e-cf04-4d80-bf93-f5753f3b0998). This example uses [TermSrvBindSecure](#Section_bc8c6536baff466bb12e435b66577bbd) from section 4.3.

1. Get the RCM binding.
2. HANDLE GetRCMBinding(LPWSTR pszServerName)
3. {
4. RPC\_STATUS rpcStatus = RPC\_S\_OK;
5. HANDLE hRCMBinding = NULL;
6. rpcStatus = TermSrvBindSecure(
7. gpszPublicUuid,
8. gpszRemoteProtocolSequence,
9. pszServerName,
10. TSRCMRPC\_REMOTE\_ENDPOINT,
11. gpszOptions,
12. &hRCMBinding
13. );
14. if( rpcStatus != RPC\_S\_OK || hRCMBinding == NULL)
15. {
16. wprintf(L"ERR: TermSrvBindSecure failed: %d\n", rpcStatus );
17. SetLastError( rpcStatus );
18. }
19. return hRCMBinding
20. }
21. Get the list of listeners.
22. hRCMBind = GetRCMBinding(pszServerName);
23. if(hRCMBind)
24. {
25. RpcTryExcept
26. {
27. hr = RpcGetAllListeners( hRCMBind, &pListeners,
28. CURRENT\_LST\_ENUM\_LEVEL, &NumListeners );
29. if(hr == S\_OK)
30. {
31. for(ULONG i=0;i<Entries;i++)
32. {
33. wprintf(L"%-10d %-20s %-40s\n",
34. pListeners[i].Data.ListenerEnum\_Level1.Id, WinstationStateClassNames[pListeners[i].Data.ListenerEnum\_Level1.bListening ? State\_Listen : State\_Down],
35. pListeners[i].Data.ListenerEnum\_Level1.Name);
36. }
37. rv2 = TRUE;
38. }
39. else
40. {
41. wprintf(L"ERR: RpcGetAllListeners failed %d\n",hr );
42. }
43. if ( pListeners )
44. {
45. MIDL\_user\_free( pListeners );
46. }
47. }
48. RpcExcept(I\_RpcExceptionFilter(RpcExceptionCode()))
49. {
50. hr = HRESULT\_FROM\_WIN32( RpcExceptionCode() );
51. wprintf(L"ERR: RpcGetAllListeners threw an exception: 0x%x\n",
52. hr );
53. }
54. RpcEndExcept
55. RpcBindingFree(&hRCMBind);
56. }
57. Close the [**handle**](#gt_5044babb-08e3-4bb9-bc12-fe8f542b05ee).
59. if(hRCMBind)
60. RpcBindingFree(&hRCMBind);

## TermSrvBindSecure Example

The following example creates an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding to an [**endpoint**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee) that uses authentication, authorization, and security quality-of-service information.

1. RPC\_STATUS
2. TermSrvBindSecure(
3. LPCWSTR pszUuid,
4. LPCWSTR pszProtocolSequence,
5. LPCWSTR pszNetworkAddress,
6. LPCWSTR pszEndPoint,
7. LPCWSTR pszOptions,
8. RPC\_BINDING\_HANDLE \*pHandle
9. )
10. {
11. RPC\_STATUS Status;
12. RPC\_SECURITY\_QOS qos;
13. LPWSTR wszServerSPN = NULL;
14. \*pHandle = NULL;
15. Status = TermSrvBind(
16. pszUuid,
17. pszProtocolSequence,
18. pszNetworkAddress,
19. pszEndPoint,
20. pszOptions,
21. pHandle);
22. if( Status != RPC\_S\_OK )
23. {
24. wprintf ( L"Error %d in TermSrvBind", Status );
25. goto TS\_EXIT\_POINT;
26. }
27. qos.Capabilities = RPC\_C\_QOS\_CAPABILITIES\_MUTUAL\_AUTH;
28. qos.IdentityTracking = RPC\_C\_QOS\_IDENTITY\_DYNAMIC;
29. qos.ImpersonationType = RPC\_C\_IMP\_LEVEL\_IMPERSONATE;
30. qos.Version = RPC\_C\_SECURITY\_QOS\_VERSION;
31. if( PrepareServerSPN( pszNetworkAddress, &wszServerSPN ))
32. {
33. Status = RpcBindingSetAuthInfoEx(
34. \*pHandle,
35. wszServerSPN,
36. RPC\_C\_AUTHN\_LEVEL\_PKT\_PRIVACY,
37. RPC\_C\_AUTHN\_GSS\_NEGOTIATE,
38. NULL,
39. RPC\_C\_AUTHZ\_NAME,
40. &qos);
41. LocalFree(wszServerSPN);
42. }
43. else
44. {
45. Status = RpcBindingSetAuthInfoEx(
46. \*pHandle,
47. (LPWSTR)pszNetworkAddress,
48. RPC\_C\_AUTHN\_LEVEL\_PKT\_PRIVACY,
49. RPC\_C\_AUTHN\_GSS\_NEGOTIATE,
50. NULL,
51. RPC\_C\_AUTHZ\_NAME,
52. &qos);
53. }
54. if ( RPC\_S\_OK != Status )
55. {
56. wprintf ( L"Error %d in RpcBindingSetAuthInfoEx", Status );
57. goto TS\_EXIT\_POINT;
58. }
59. TS\_EXIT\_POINT:
60. if ( RPC\_S\_OK != Status &&
61. NULL != \*pHandle )
62. {
63. RpcBindingFree( pHandle );
64. }
65. return Status;
66. }

Generate a standard RPC binding from the protocol sequence, security options, and [**UUID**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3), for example.

1. RPC\_STATUS
2. TermSrvBind(
3. IN LPCWSTR pszUuid,
4. IN LPCWSTR pszProtocolSequence,
5. IN LPCWSTR pszNetworkAddress,
6. IN LPCWSTR pszEndPoint,
7. IN LPCWSTR pszOptions,
8. OUT RPC\_BINDING\_HANDLE \*pHandle
9. )
10. {
11. RPC\_STATUS Status;
12. LPWSTR pszString = NULL;
13. /\*
14. \* Compose the binding string using the helper routine
15. \* and our protocol sequence, security options, UUID, and so on.
16. \*/
17. Status = RpcStringBindingCompose(
18. (LPWSTR)pszUuid,
19. (LPWSTR)pszProtocolSequence,
20. (LPWSTR)pszNetworkAddress,
21. (LPWSTR)pszEndPoint,
22. (LPWSTR)pszOptions,
23. &pszString
24. );
25. if( Status != RPC\_S\_OK )
26. {
27. wprintf ( L"Error %d in RpcStringBindingCompose", Status );
28. goto TS\_EXIT\_POINT;
29. }
30. /\*
31. \* Now generate the RPC binding from the canonical RPC
32. \* binding string.
33. \*/
34. Status = RpcBindingFromStringBinding(
35. pszString,
36. pHandle
37. );
38. if( Status != RPC\_S\_OK )
39. {
40. wprintf ( L"Error %d in RpcBindingFromStringBinding", Status );
41. goto TS\_EXIT\_POINT;
42. }
43. TS\_EXIT\_POINT:
44. if ( NULL != pszString )
45. {
46. /\*
47. \* Free the memory returned from RpcStringBindingCompose()
48. \*/
49. RpcStringFree( &pszString );
50. }
51. return( Status );
52. }

Recreate a valid SPN for Windows Vista operating system from an existing SPN.

1. BOOL
2. PrepareServerSPN(
3. IN LPCWSTR pszNetworkAddress,
4. \_\_deref\_out\_opt LPWSTR \*ppwszServerSPN
5. )
6. {
7. // Windows Server 2008 RPC does not accept "net use" credential anymore.
8. // <Domain>\<Machine> is not a valid SPN, a valid SPN is host/<Machine Name>
9. LPWSTR pszTemplate = L"host/%s";
10. \*ppwszServerSPN = NULL;
11. HRESULT hr = S\_OK;
12. UINT stringLength = wcslen(pszTemplate)+wcslen(pszNetworkAddress)+1;
13. \*ppwszServerSPN = (LPWSTR)LocalAlloc(LPTR, stringLength \* sizeof(WCHAR));
14. if(\*ppwszServerSPN)
15. {
16. hr = StringCchPrintf( \*ppwszServerSPN, stringLength, pszTemplate,
17. pszNetworkAddress );
18. ASSERT( SUCCEEDED( hr ));
19. }
20. if( FAILED(hr) )
21. {
22. if( NULL != \*ppwszServerSPN )
23. {
24. LocalFree( \*ppwszServerSPN );
25. \*ppwszServerSPN = NULL;
26. }
27. }
28. return SUCCEEDED(hr);
29. }

## Legacy Example

The following example shows how to enumerate sessions that use the legacy [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) methods.

1. Get the binding.
2. Result = TermSrvBindSecure(
3. pszUuid,
4. pszRemoteProtocolSequence,
5. pServerName,
6. pszRemoteEndPoint,
7. pszOptions,
8. &RpcTSHandle
9. );
10. //
11. // Get a context handle from the server so it can
12. // manage the connections state
13. //
14. RpcTryExcept {
15. rc = RpcWinStationOpenServer( RpcTSHandle, &Result, &ContextHandle );
16. }
17. RpcExcept(I\_RpcExceptionFilter(RpcExceptionCode())) {
18. Result = RpcExceptionCode();
19. rc = FALSE;
20. wprintf(L"ERR: RPC Exception %d\n",Result );
21. }
22. RpcEndExcept
23. Enumerate the sessions.
24. RpcTryExcept {
25. rc = RpcWinStationEnumerate(
26. hServer,
27. &Result,
28. &LogonIdCount,
29. (PCHAR)pLogonId,
30. &Length,
31. &Index
32. );
33. Result = RtlNtStatusToDosError( Result );
34. if ( Result == ERROR\_NO\_MORE\_ITEMS) {
35. Result = ERROR\_SUCCESS;
36. break;
37. }
38. if(rc == TRUE)
39. {
40. wprintf(L"SessionID State Name\n");
41. for(ULONG i=0;i<LogonIdCount;i++)
42. {
43. wprintf(L"%-10d %-20s %-40s\n",
44. pLogonId[i].SessionId,
45. WinstationStateClassNames[pLogonId[i].State],
46. pLogonId[i].WinStationName);
47. }
48. }
49. }
50. RpcExcept(I\_RpcExceptionFilter(RpcExceptionCode())) {
51. Result = RpcExceptionCode();
52. wprintf(L"ERR: RPC Exception %d\n",Result );
53. }
54. RpcEndExcept
55. Close the binding handles.
56. RpcTryExcept {
57. bSuccess = RpcWinStationCloseServerEx( pHandle, pdwResult );
58. if( !bSuccess ) \*pdwResult = RtlNtStatusToDosError( \*pdwResult );
59. }
60. RpcExcept(I\_RpcExceptionFilter(RpcExceptionCode())) {
61. \*pdwResult = RpcExceptionCode();
62. bSuccess = FALSE;
63. }
64. RpcEndExcept

The following diagram illustrates the message sequence for enumerating the sessions.

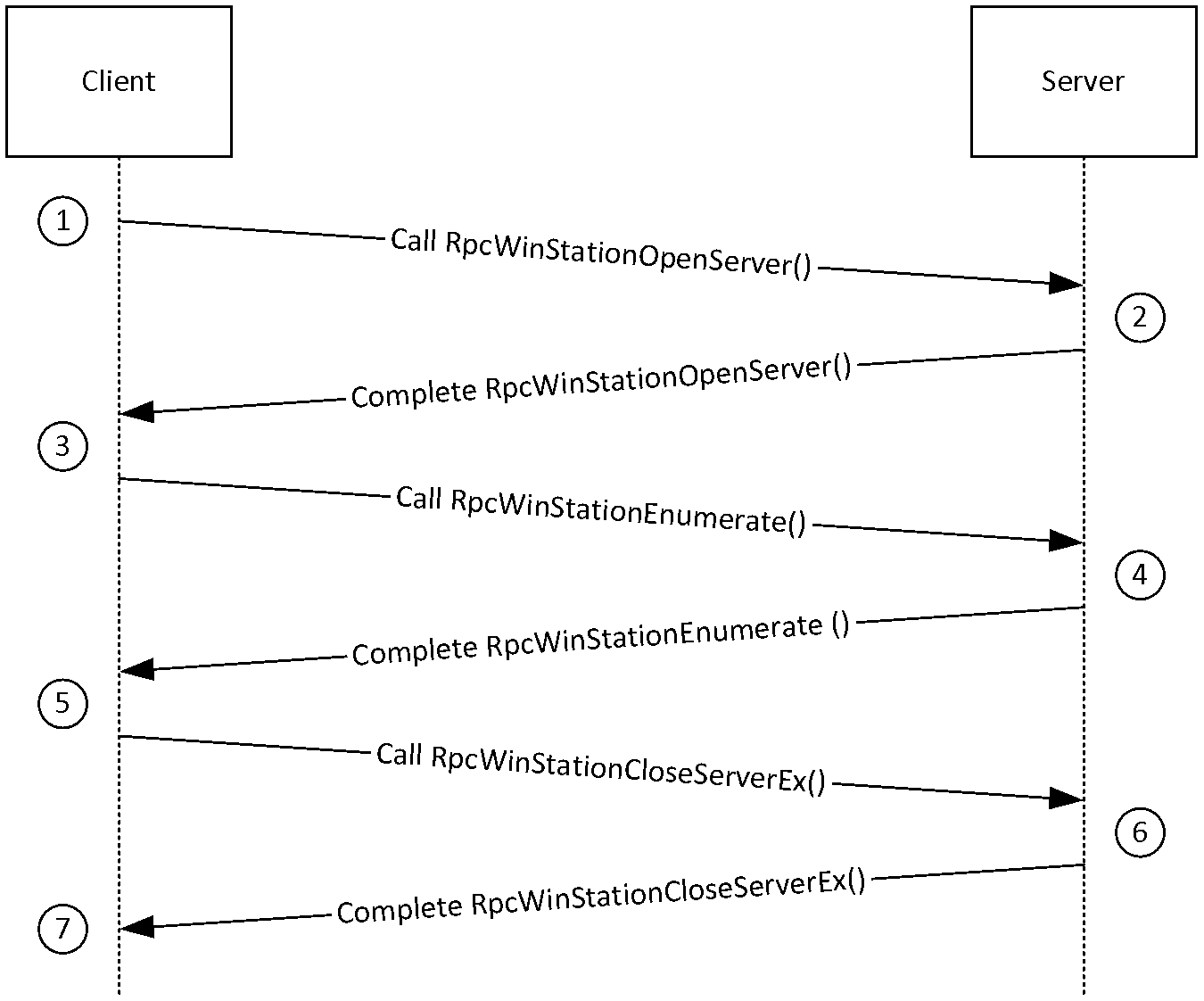


Figure 2: Legacy session enumeration sequence

The sequence of messages for enumerating sessions on the server is as follows:

1. After an RPC binding has been established to the server, the client requests a handle to the server to be opened by calling the [RpcWinStationOpenServer](#Section_fd6f339df5dd452ca9a9ec753c22a36a) method.
2. The server in response will open up a handle and return to the client.
3. The client then calls the [RpcWinStationEnumerate](#Section_fa654a9ca67f4e589fa75dd0ae6305a1) method by passing this handle along with an uninitialized buffer to get the list of sessions.
4. The server on receiving the request will allocate memory for the buffer and fill it with an array of [LOGONID](#Section_e5c755e86d66494d873f4540267fe2f4) structures containing session information, one for each session on the server. It will also return the number of sessions on the server.
5. The client, on receiving the data, calls the [RpcWinStationCloseServerEx](#Section_9d198d98b7f1475d8c29cc188f5bc0fc) method to inform the server to close the server handle.
6. The server on receiving the RpcWinStationCloseServerEx call will close the server handle.
7. The client frees the array of [SESSIONENUM](#Section_fd8759ef4f6c45c6b5429d5e53e90f7d) structures it received before exiting.

## Encoding/Decoding Example

The following is the example of encoding and decoding the PropValue field in the [TSProperty](#Section_0bd1a91d3cd64e97b5fa5cf35d90f005) structure.

1. #include <stdio.h>
2. #include <tchar.h>
3. #include <Windows.h>
4. DWORD
5. EncodePropValue
6. (
7. \_\_in BYTE\* pbSource,
8. \_\_in DWORD dwSourceLength,
9. \_\_deref\_out\_bcount(\*pdwDestLength) BYTE\*\* ppbDest,
10. \_\_out DWORD\* pdwDestLength
11. );
12. DWORD
13. DecodePropValue
14. (
15. \_\_in BYTE\* pbSource,
16. \_\_in DWORD dwSourceLength,
17. \_\_deref\_out\_bcount(\*pdwDestLength) BYTE\*\* ppbDest,
18. DWORD \*pdwDestLength
19. );
20. int \_tmain()
21. {
23. char\* pPropValue = "ABCDE";
24. char\* pEncoded = NULL;
25. DWORD cbEncoded = 0;
26. char\* pDecoded = NULL;
27. DWORD cbDecoded = 0;
28. //
29. // Encoding a property value to be compatible with TSProperty structure.
30. //
31. EncodePropValue( PBYTE(pPropValue), (strlen(pPropValue)+1), (PBYTE\*)&pEncoded, &cbEncoded );
32. //
33. // Decoding the encoded string.
34. //
35. DecodePropValue( PBYTE(pEncoded), cbEncoded, (PBYTE\*)&pDecoded, &cbDecoded );
36. printf("Decoded: %s\n", pDecoded);
37. delete[] PBYTE(pEncoded);
38. delete[] PBYTE(pDecoded);
39. return 0;
40. }
41. DWORD
42. EncodePropValue
43. (
44. \_\_in BYTE\* pbSource,
45. \_\_in DWORD dwSourceLength,
46. \_\_deref\_out\_bcount(\*pdwDestLength) BYTE\*\* ppbDest,
47. \_\_out DWORD\* pdwDestLength
48. )
49. {
50. \*ppbDest = new BYTE[(dwSourceLength\*2)+1];
51. for(DWORD i=0; i<dwSourceLength; i++)
52. {
53. sprintf((char\*)((\*ppbDest)+(i\*2)), "%02x", pbSource[i]);
54. }
56. \*pdwDestLength = dwSourceLength\*2;
57. return 0;
58. }
59. #define MAPHEXTODIGIT(x) ( x >= '0' && x <= '9' ? (x-'0') : \
60. x >= 'A' && x <= 'F' ? (x-'A'+10) : \
61. x >= 'a' && x <= 'f' ? (x-'a'+10) : 0 )
62. DWORD
63. DecodePropValue
64. (
65. \_\_in BYTE\* pbSource,
66. \_\_in DWORD dwSourceLength,
67. \_\_deref\_out\_bcount(\*pdwDestLength) BYTE\*\* ppbDest,
68. DWORD \*pdwDestLength
69. )
70. {
71. \*pdwDestLength = dwSourceLength/2;
72. (\*ppbDest) = new BYTE[\*pdwDestLength];
73. for(DWORD i=0; i<(\*pdwDestLength); i++)
74. {
75. (\*ppbDest)[i] = MAPHEXTODIGIT( pbSource[2\*i]) \* 16 +
76. MAPHEXTODIGIT( pbSource[2\*i+1]);
77. }
78. return 0;
79. }

# Security

## Security Considerations for Implementers

The Terminal Services Terminal Server Runtime Interface Protocol allows any user to connect to the server, as specified in section [2.1](#Section_ac0535d0a8474386ae3ee93a50ff2484). Therefore, any security bug in the server implementation could be exploitable. The server implementation enforces security on each method.

## Index of Security Parameters

The only security parameter is Authentication Protocol, section [2.1](#Section_ac0535d0a8474386ae3ee93a50ff2484).

# Appendix A: Full IDL

For ease of implementation, the full [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) and headers are provided in the following sections, where "ms-dtyp.idl" is the IDL as described in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) Appendix A.

When compiling these IDLs with the MIDL compiler, "MIDL\_PASS" has to be defined. This can be done by using the midl command line with the /D switch. For example, "midl /D MIDL\_PASS legacy.idl".

## Appendix A.1: tspubrpc.idl

For ease of implementation, the full IDL is provided, where "ms-dtyp.idl" is the IDL as described in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) Appendix A and "tsdef.h" is as specified in section [6.5](#Section_ce70794f213843e8bf6c2c147887d6a2).

1. import "ms-dtyp.idl";
2. #include "ms-tsts\_tsdef.h"
3. #include "ms-tsts\_allproc.h"
4. [
5. uuid(484809d6-4239-471b-b5bc-61df8c23ac48),
6. version(1.0),
7. pointer\_default(unique)
8. ]
9. //
10. // Public rpc interface to the session object
11. //
12. interface TermSrvSession
13. {
14. cpp\_quote("#define WTS\_SESSIONSTATE\_UNKNOWN 0xFFFFFFFF")
15. cpp\_quote("#define WTS\_SESSIONSTATE\_LOCK 0x00000000")
16. cpp\_quote("#define WTS\_SESSIONSTATE\_UNLOCK 0x00000001")
17. typedef [context\_handle] void \*SESSION\_HANDLE;
18. typedef struct \_LSMSessionInformation {
19. [string] WCHAR\* pszUserName;
20. [string] WCHAR\* pszDomain;
21. [string] WCHAR\* pszTerminalName;
22. LONG SessionState;
23. BOOL DesktopLocked;
24. hyper ConnectTime;
25. hyper DisconnectTime;
26. hyper LogonTime;
27. } LSMSESSIONINFORMATION,
28. \*PLSMSESSIONINFORMATION;
29. typedef struct \_LSM\_SESSIONINFO\_EX\_LEVEL1 {
30. LONG SessionState;
31. LONG SessionFlags;
32. WCHAR SessionName[33];
33. WCHAR DomainName[18];
34. WCHAR UserName[21];
35. hyper ConnectTime;
36. hyper DisconnectTime;
37. hyper LogonTime;
38. hyper LastInputTime;
39. ULONG ProtocolDataSize;
40. [size\_is(ProtocolDataSize)] PBYTE ProtocolData;
41. } LSM\_SESSIONINFO\_EX\_LEVEL1,
42. \*PLSM\_SESSIONINFO\_EX\_LEVEL1;
43. typedef [switch\_type (DWORD)] union \_LSM\_SESSIONINFO\_EX {
44. [case(1)]
45. LSM\_SESSIONINFO\_EX\_LEVEL1 LSM\_SessionInfo\_Level1;
46. } LSM\_SESSIONINFO\_EX,
47. \*PLSM\_SESSIONINFO\_EX;
48. typedef struct \_LSMSESSIONINFORMATION\_EX {
49. DWORD Level;
50. [switch\_is(Level)] LSM\_SESSIONINFO\_EX Data;
51. } LSMSESSIONINFORMATION\_EX,
52. \*PLSMSESSIONINFORMATION\_EX;
53. //
54. // Per Session specific call
55. HRESULT RpcOpenSession(
56. [in] handle\_t hBinding,
57. [in] LONG SessionId,
58. [out] SESSION\_HANDLE \*phSession
59. );
60. HRESULT RpcCloseSession(
61. [in,out] SESSION\_HANDLE \*phSession
62. );
63. HRESULT RpcConnect(
64. [in] SESSION\_HANDLE hSession,
65. [in] LONG TargetSessionId,
66. [in,string] WCHAR \*szPassword
67. );
68. HRESULT RpcDisconnect(
69. [in] SESSION\_HANDLE hSession
70. );
71. HRESULT RpcLogoff(
72. [in] SESSION\_HANDLE hSession
73. );
74. HRESULT RpcGetUserName(
75. [in] SESSION\_HANDLE hSession,
76. [out, string] WCHAR \*\*pszUserName,
77. [out, string] WCHAR \*\*pszDomain
78. );
79. HRESULT RpcGetTerminalName(
80. [in] SESSION\_HANDLE hSession,
81. [out, string] WCHAR \*\*pszTerminalName
82. );
83. HRESULT RpcGetState(
84. [in] SESSION\_HANDLE hSession,
85. [out] LONG \*plState
86. );
87. HRESULT RpcIsSessionDesktopLocked(
88. [in] SESSION\_HANDLE hSession
89. );
90. HRESULT RpcShowMessageBox(
91. [in] SESSION\_HANDLE hSession,
92. [in, string] WCHAR \*szTitle,
93. [in, string] WCHAR \*szMessage,
94. [in] ULONG ulStyle,
95. [in] ULONG ulTimeout,
96. [out] ULONG \*pulResponse,
97. [in] BOOL bDoNotWait
98. );
99. HRESULT RpcGetTimes(
100. [in] SESSION\_HANDLE hSession,
101. [out] hyper \*pConnectTime,
102. [out] hyper \*pDisconnectTime,
103. [out] hyper \*pLogonTime
104. );
105. HRESULT RpcGetSessionCounters(
106. [in] handle\_t hBinding,
107. [in,out, size\_is(uEntries)] PTS\_COUNTER pCounter,
108. [in] ULONG uEntries
109. );
110. HRESULT RpcGetSessionInformation(
111. [in] handle\_t hBinding,
112. [in] LONG SessionId,
113. [ref, out] PLSMSESSIONINFORMATION pSessionInfo
114. );
115. VOID Opnum13NotUsedOnWire();
116. VOID Opnum14NotUsedOnWire();
117. HRESULT RpcGetLoggedOnCount(
118. [in] handle\_t hBinding,
119. [out] ULONG \*pUserSessions,
120. [out] ULONG \*pDeviceSessions
121. );
122. HRESULT RpcGetSessionType (
123. [in] handle\_t hBinding,
124. [in] LONG \*SessionId,
125. [out] ULONG \*pSessionType
126. );
127. HRESULT RpcGetSessionInformationEx(
128. [in] handle\_t hBinding,
129. [in] LONG SessionId,
130. [in] DWORD Level,
131. [ref, out] PLSMSESSIONINFORMATION\_EX LSMSessionInfoExPtr
132. );
133. }
135. VOID Opnum18NotUsedOnWire();
136. VOID Opnum19NotUsedOnWire();
137. //
138. // notifications
139. //
140. [
141. uuid(11899a43-2b68-4a76-92e3-a3d6ad8c26ce),
142. version(1.0),
143. pointer\_default(unique)
144. ]
145. interface TermSrvNotification
146. {
147. HRESULT RpcWaitForSessionState(
148. [in] handle\_t hBinding,
149. [in] LONG SessionId,
150. [in] LONG State,
151. [in] ULONG Timeout
152. );
153. typedef [context\_handle] void \*NOTIFY\_HANDLE;
154. HRESULT RpcRegisterAsyncNotification(
155. [in] handle\_t hBinding,
156. [in] LONG SessionId,
157. [in] TNotificationId Mask,
158. [out] NOTIFY\_HANDLE \*phNotify
159. );
160. HRESULT RpcWaitAsyncNotification(
161. [in] NOTIFY\_HANDLE hNotify,
162. [out, size\_is(, \*pEntries)]
163. PSESSION\_CHANGE \*SessionChange,
164. [out] ULONG \*pEntries
165. );
166. HRESULT RpcUnRegisterAsyncNotification(
167. [in,out] NOTIFY\_HANDLE \*phNotify
168. );
169. }
170. //
171. // enumerations
172. //
173. [
174. uuid(88143fd0-c28d-4b2b-8fef-8d882f6a9390),
175. version(1.0),
176. pointer\_default(unique)
177. ]
178. interface TermSrvEnumeration
179. {
180. typedef [context\_handle] void \*ENUM\_HANDLE;
181. #define ENUM\_LEVEL1 1
182. #define ENUM\_LEVEL2 2
183. #define ENUM\_LEVEL3 3
184. #define CURRENT\_ENUM\_LEVEL 2
185. typedef struct \_SESSIONENUM\_LEVEL1 {
186. LONG SessionId;
187. LONG State;
188. WCHAR Name[33];
189. } SESSIONENUM\_LEVEL1,
190. \*PSESSIONENUM\_LEVEL1;
191. typedef struct \_SESSIONENUM\_LEVEL2 {
192. LONG SessionId;
193. LONG State;
194. WCHAR Name[33];
195. ULONG Source;
196. BOOL bFullDesktop;
197. GUID SessionType;
198. } SESSIONENUM\_LEVEL2,
199. \*PSESSIONENUM\_LEVEL2;
200. typedef struct \_SESSIONENUM\_LEVEL3 {
201. LONG SessionId;
202. LONG State;
203. WCHAR Name[33];
204. ULONG Source;
205. BOOL bFullDesktop;
206. GUID SessionType;
207. ULONG ProtoDataSize;
208. [size\_is(ProtoDataSize)]UCHAR \* pProtocolData;
209. } SESSIONENUM\_LEVEL3,
210. \*PSESSIONENUM\_LEVEL3;
211. typedef [switch\_type(DWORD)] union \_SessionInfo {
212. [case(1)]
213. SESSIONENUM\_LEVEL1 SessionEnum\_Level1;
214. [case(2)]
215. SESSIONENUM\_LEVEL2 SessionEnum\_Level2;
216. } SessionInfo,
217. \*PSessionInfo;
218. typedef struct \_SESSIONENUM {
219. DWORD Level;
220. [switch\_is(Level)] SessionInfo Data;
221. } SESSIONENUM,
222. \*PSESSIONENUM;
223. typedef [switch\_type(DWORD)] union \_SessionInfo\_Ex {
224. [case(1)]
225. SESSIONENUM\_LEVEL1 SessionEnum\_Level1;
226. [case(2)]
227. SESSIONENUM\_LEVEL2 SessionEnum\_Level2;
228. [case(3)]
229. SESSIONENUM\_LEVEL3 SessionEnum\_Level3;
230. } SessionInfo\_Ex,
231. \*PSessionInfo\_Ex;
232. typedef struct \_SESSIONENUM\_EX {
233. DWORD Level;
234. [switch\_is(Level)] SessionInfo\_Ex Data;
235. } SESSIONENUM\_EX,
236. \*PSESSIONENUM\_EX;
237. #define UNIFIED\_ENUM\_LEVEL1 1
238. #define UNIFIED\_ENUM\_LEVEL2 2
239. #define CURRENT\_UNIFIED\_ENUM\_LEVEL 2
240. typedef struct \_EXECENVDATA\_LEVEL1 {
241. LONG ExecEnvId;
242. LONG State;
243. WCHAR SessionName[33];
244. } EXECENVDATA\_LEVEL1,
245. \*PEXECENVDATA\_LEVEL1;
246. typedef struct \_EXECENVDATA\_LEVEL2 {
247. LONG ExecEnvId;
248. LONG State;
249. WCHAR SessionName[33];
250. LONG AbsSessionId;
251. WCHAR HostName[33];
252. WCHAR UserName[33];
253. WCHAR DomainName[33];
254. WCHAR FarmName[33];
255. } EXECENVDATA\_LEVEL2,
256. \*PEXECENVDATA\_LEVEL2;
257. typedef [switch\_type(DWORD)] union \_ExecEnvData {
258. [case(1)]
259. EXECENVDATA\_LEVEL1 ExecEnvEnum\_Level1;
260. [case(2)]
261. EXECENVDATA\_LEVEL2 ExecEnvEnum\_Level2;
262. } ExecEnvData,
263. \*PExecEnvData;
264. typedef struct \_EXECENVDATA {
265. DWORD Level;
266. [switch\_is(Level)] ExecEnvData Data;
267. } EXECENVDATA,
268. \*PEXECENVDATA;
269. typedef struct \_EXECENVDATAEX\_LEVEL1 {
270. LONG ExecEnvId;
271. LONG State;
272. LONG AbsSessionId;
273. [string, max\_is(256)] LPWSTR pszSessionName;
274. [string, max\_is(256)] LPWSTR pszHostName;
275. [string, max\_is(256)] LPWSTR pszUserName;
276. [string, max\_is(256)] LPWSTR pszDomainName;
277. [string, max\_is(256)] LPWSTR pszFarmName;
278. } EXECENVDATAEX\_LEVEL1,
279. \*PEXECENVDATAEX\_LEVEL1;
280. typedef [switch\_type(DWORD)] union \_ExecEnvDataEx {
281. [case(1)]
282. EXECENVDATAEX\_LEVEL1 ExecEnvEnum\_Level1;
283. } ExecEnvDataEx,
284. \*PExecEnvDataEx;
285. typedef struct \_EXECENVDATAEX {
286. DWORD Level;
287. [switch\_is(Level)] ExecEnvDataEx Data;
288. } EXECENVDATAEX,
289. \*PEXECENVDATAEX;
290. HRESULT RpcOpenEnum(
291. [in] handle\_t hBinding,
292. [out] ENUM\_HANDLE \*phEnum
293. );
294. HRESULT RpcCloseEnum(
295. [in,out] ENUM\_HANDLE \*phEnum
296. );
297. HRESULT RpcFilterByState(
298. [in] ENUM\_HANDLE hEnum,
299. [in] LONG State,
300. [in] BOOL bInvert
301. );
302. HRESULT RpcFilterByCallersName(
303. [in] ENUM\_HANDLE hEnum
304. );
305. HRESULT RpcEnumAddFilter(
306. [in] ENUM\_HANDLE hEnum,
307. [in] ENUM\_HANDLE hSubEnum
308. );
309. HRESULT RpcGetEnumResult(
310. [in] ENUM\_HANDLE hEnum,
311. [out, size\_is(,\*pEntries)]
312. PSESSIONENUM \*ppSessionEnumResult,
313. [in] DWORD Level,
314. [out] ULONG \*pEntries
315. );
316. HRESULT RpcFilterBySessionType(
317. [in] ENUM\_HANDLE hEnum,
318. [in] GUID\* pSessionType
319. );
320. VOID Opnum7NotUsedOnWire(void);
321. HRESULT RpcGetSessionIds(
322. [in] handle\_t hBinding,
323. [in] SESSION\_FILTER Filter,
324. [in, range( 0, 0xFFFF )]
325. ULONG MaxEntries,
326. [out, size\_is(,\*pcSessionIds)]
327. LONG\*\* pSessionIds,
328. [out] ULONG\* pcSessionIds
329. );
330. HRESULT RpcGetEnumResultEx(
331. [in] ENUM\_HANDLE hEnum,
332. [out, size\_is(,\*pEntries)]
333. PSESSIONENUM\_EX \*ppSessionEnumResult,
334. [in] DWORD Level,
335. [out] ULONG \*pEntries
336. );
337. HRESULT RpcGetAllSessions(
338. [in] handle\_t hBinding,
339. [in, out] ULONG \*pLevel,
340. [out, size\_is(,\*pcEntries)]
341. PEXECENVDATA \*ppSessionData,
342. [out] ULONG \*pcEntries
343. );
344. HRESULT RpcGetAllSessionsEx(
345. [in] handle\_t hBinding,
346. [in] ULONG Level,
347. [out, size\_is(,\*pcEntries)]
348. PEXECENVDATAEX \*ppSessionData,
349. [out] ULONG \*pcEntries
350. );
351. }

## Appendix A.2: rcmpublic.idl

For ease of implementation, the full [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) is provided, where "ms-dtyp.idl" is the IDL as described in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) Appendix A.

1. import "ms-dtyp.idl";
2. //
3. // public access ( local & remote )
4. //
5. [
6. uuid(bde95fdf-eee0-45de-9e12-e5a61cd0d4fe),
7. version(1.0),
8. pointer\_default(unique)
9. ]
10. interface RCMPublic
11. {
12. static const WCHAR TSRCMRPC\_REMOTE\_ENDPOINT[] = L"\\pipe\\TermSrv\_API\_service";
13. typedef struct {
14. USHORT sin\_family;
15. union switch (USHORT sin\_family) {
16. case 2:
17. struct {
18. USHORT sin\_port;
19. ULONG in\_addr;
20. UCHAR sin\_zero[8];
21. } ipv4;
22. case 23:
23. struct {
24. USHORT sin6\_port;
25. ULONG sin6\_flowinfo;
26. USHORT sin6\_addr[8];
27. ULONG sin6\_scope\_id;
28. } ipv6;
29. };
30. } RCM\_REMOTEADDRESS, \*PRCM\_REMOTEADDRESS;
31. typedef WCHAR LISTENER\_NAME[32];
32. #define LST\_ENUM\_LEVEL1 1
33. #define CURRENT\_LST\_ENUM\_LEVEL 1
34. typedef struct \_LISTENERENUM\_LEVEL1 {
35. LONG Id;
36. BOOL bListening;
37. WCHAR Name[33];
38. } LISTENERENUM\_LEVEL1, \*PLISTENERENUM\_LEVEL1;
39. typedef [switch\_type(DWORD)] union \_ListenerInfo {
40. [case(1)]
41. LISTENERENUM\_LEVEL1 ListenerEnum\_Level1;
42. [default]
43. ;
44. } ListenerInfo, \*PListenerInfo;
45. typedef struct \_LISTENERENUM {
46. DWORD Level;
47. [switch\_is(Level)] ListenerInfo Data;
48. } LISTENERENUM, \*PLISTENERENUM;
49. HRESULT RpcGetClientData(
50. [in] handle\_t hBinding,
51. [in] ULONG SessionId,
52. [out, size\_is( ,\*pOutBuffByteLen )]
53. unsigned char \*\*ppBuff,
54. [out] ULONG \*pOutBuffByteLen
55. );
56. HRESULT RpcGetConfigData(
57. [in] handle\_t hBinding,
58. [in] ULONG SessionId,
59. [out, size\_is( ,\*pOutBuffByteLen )]
60. unsigned char \*\*ppBuff,
61. [out] ULONG \*pOutBuffByteLen
62. );
63. typedef enum {
64. PROTOCOLSTATUS\_INFO\_BASIC = 0,
65. PROTOCOLSTATUS\_INFO\_EXTENDED = 1,
66. } PROTOCOLSTATUS\_INFO\_TYPE;
67. HRESULT RpcGetProtocolStatus(
68. [in] handle\_t hBinding,
69. [in] ULONG SessionId,
70. [in] PROTOCOLSTATUS\_INFO\_TYPE InfoType,
71. [out, size\_is( ,\*pcbProtoStatus )]
72. unsigned char \*\*ppProtoStatus,
73. [out] ULONG \*pcbProtoStatus
74. );
75. HRESULT RpcGetLastInputTime(
76. [in] handle\_t hBinding,
77. [in] ULONG SessionId,
78. [out] hyper \*pLastInputTime
79. );
80. HRESULT RpcGetRemoteAddress(
81. [in] handle\_t hBinding,
82. [in] ULONG SessionId,
83. [out] PRCM\_REMOTEADDRESS pRemoteAddress
84. );
85. VOID Opnum5NotUsedOnWire();
86. VOID Opnum6NotUsedOnWire();
87. VOID Opnum7NotUsedOnWire();
88. HRESULT RpcGetAllListeners(
89. [in] handle\_t hBinding,
90. [out, size\_is(,\*pNumListeners)] PLISTENERENUM \*ppListeners,
91. [in] DWORD Level,
92. [out] ULONG \*pNumListeners
93. );
94. HRESULT RpcGetSessionProtocolLastInputTime(
95. [in] handle\_t hBinding,
96. [in] ULONG SessionId,
97. [in] PROTOCOLSTATUS\_INFO\_TYPE InfoType,
98. [out, size\_is( ,\*pcbProtoStatus )]
99. unsigned char \*\*ppProtoStatus,
100. [out] ULONG \*pcbProtoStatus,
101. [out] hyper \*pLastInputTime
102. );
103. HRESULT RpcGetUserCertificates(
104. [in] handle\_t hBinding,
105. [in] ULONG SessionId,
106. [out] ULONG\* pcCerts,
107. [out, size\_is(, \*pcbCerts)] byte\*\* ppbCerts,
108. [out] ULONG\* pcbCerts
109. );
110. typedef enum {
111. QUERY\_SESSION\_DATA\_MODULE = 0,
112. QUERY\_SESSION\_DATA\_WDCONFIG,
113. QUERY\_SESSION\_DATA\_VIRTUALDATA,
114. QUERY\_SESSION\_DATA\_LICENSE,
115. QUERY\_SESSION\_DATA\_DEVICEID,
116. QUERY\_SESSION\_DATA\_LICENSE\_VALIDATION
117. } QUERY\_SESSION\_DATA\_TYPE;
118. HRESULT RpcQuerySessionData(
119. [in] handle\_t hBinding,
120. [in] ULONG SessionId,
121. [in] QUERY\_SESSION\_DATA\_TYPE type,
122. [in, unique, size\_is( cbInputData )] byte\* pbInputData,
123. [in, range(0, 8192) ] DWORD cbInputData,
124. [out, ref, size\_is(cbSessionData), length\_is(\*pcbReturnLength)]
125. byte\* pbSessionData,
126. [in, range(0, 8192)] ULONG cbSessionData,
127. [out, ref] ULONG \*pcbReturnLength,
128. [out, ref] ULONG \*pcbRequireBufferSize
129. );
130. };
131. //
132. // Describe a listener
133. //
134. [
135. uuid(497d95a6-2d27-4bf5-9bbd-a6046957133c),
136. version(1.0),
137. pointer\_default(unique)
138. ]
139. interface RCMListener
140. {
141. typedef [context\_handle] void \*HLISTENER;
142. HRESULT RpcOpenListener(
143. [in] handle\_t hBinding,
144. [in,string] WCHAR \*szListenerName,
145. [out] HLISTENER \*phListener
146. );
147. HRESULT RpcCloseListener(
148. [in,out] HLISTENER \*phListener
149. );
150. HRESULT RpcStopListener(
151. [in] HLISTENER hListener
152. );
153. HRESULT RpcStartListener(
154. [in] HLISTENER hListener
155. );
156. HRESULT RpcIsListening(
157. [in] HLISTENER hListener,
158. [out] BOOL \*pbIsListening
159. );
160. }

## Appendix A.3: legacy.idl

For ease of implementation, the full [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) is provided, where "ms-dtyp.idl" is the IDL as described in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) Appendix A and "allproc.h" is as specified in section [6.7](#Section_7071a3e1723c46659b387820d1ccfbda).

1. import "ms-dtyp.idl";
2. #include "ms-tsts\_allproc.h"
3. typedef UINT \* UINT\_PTR;
4. static const WCHAR TSRPC\_LEGACY\_REMOTE\_ENDPOINT[] =
5. L"\\pipe\\Ctx\_WinStation\_API\_service";
6. typedef [context\_handle] void \*SERVER\_HANDLE;
7. [
8. uuid(5ca4a760-ebb1-11cf-8611-00a0245420ed),
9. version(1.0),
10. pointer\_default(unique)
11. ]
12. interface IcaApi
13. {
14. typedef [range(0, 0x8000)] ULONG BOUNDED\_ULONG;
15. BOOLEAN RpcWinStationOpenServer(
16. [in] handle\_t hBinding,
17. [out] DWORD \*pResult,
18. [out] SERVER\_HANDLE \*phServer
19. );
20. BOOLEAN RpcWinStationCloseServer(
21. [in] SERVER\_HANDLE hServer,
22. [out] DWORD \*pResult
23. );
24. BOOLEAN RpcIcaServerPing(
25. [in] SERVER\_HANDLE hServer,
26. [out] DWORD \*pResult
27. );
28. BOOLEAN RpcWinStationEnumerate(
29. [in] SERVER\_HANDLE hServer,
30. [out] DWORD \*pResult,
31. [in,out] PULONG pEntries,
32. [in,out,unique,size\_is(\*pByteCount)]
33. PCHAR pLogonId,
34. [in,out] PULONG pByteCount,
35. [in,out] PULONG pIndex
36. );
37. BOOLEAN RpcWinStationRename(
38. [in] SERVER\_HANDLE hServer,
39. [out] DWORD \*pResult,
40. [in,size\_is(NameOldSize)]
41. PWCHAR pWinStationNameOld,
42. [in,range(0, 256)]
43. DWORD NameOldSize,
44. [in,size\_is(NameNewSize)]
45. PWCHAR pWinStationNameNew,
46. [in,range(0, 256)]
47. DWORD NameNewSize
48. );
49. BOOLEAN RpcWinStationQueryInformation(
50. [in] SERVER\_HANDLE hServer,
51. [out] DWORD \*pResult,
52. [in] DWORD LogonId,
53. [in] DWORD WinStationInformationClass,
54. [in,out,unique,size\_is(WinStationInformationLength)]
55. PCHAR pWinStationInformation,
56. [in, range( 0, 0x8000)]
57. DWORD WinStationInformationLength,
58. [out] DWORD \*pReturnLength
59. );
60. BOOLEAN RpcWinStationSetInformation(
61. [in] SERVER\_HANDLE hServer,
62. [out] DWORD \*pResult,
63. [in] DWORD LogonId,
64. [in] DWORD WinStationInformationClass,
65. [in,out,unique,size\_is(WinStationInformationLength)]
66. PCHAR pWinStationInformation,
67. [in, range( 0, 0x8000)]
68. DWORD WinStationInformationLength
69. );
70. BOOLEAN RpcWinStationSendMessage(
71. [in] SERVER\_HANDLE hServer,
72. [out] DWORD \*pResult,
73. [in] DWORD LogonId,
74. [in,size\_is(TitleLength)] PWCHAR pTitle,
75. [in, range( 0, 1024 )]
76. DWORD TitleLength,
77. [in,size\_is(MessageLength)]
78. PWCHAR pMessage,
79. [in, range( 0, 1024 )]
80. DWORD MessageLength,
81. [in] DWORD Style,
82. [in] DWORD Timeout,
83. [out] DWORD \*pResponse,
84. [in] BOOLEAN DoNotWait
85. );
86. BOOLEAN RpcLogonIdFromWinStationName(
87. [in] SERVER\_HANDLE hServer,
88. [out] DWORD \*pResult,
89. [in,size\_is(NameSize)]
90. PWCHAR pWinStationName,
91. [in, range( 0, 256 )]
92. DWORD NameSize,
93. [out] DWORD \*pLogonId
94. );
95. BOOLEAN RpcWinStationNameFromLogonId(
96. [in] SERVER\_HANDLE hServer,
97. [out] DWORD \*pResult,
98. [in] DWORD LoginId,
99. [in,out,size\_is(NameSize)]
100. PWCHAR pWinStationName,
101. [in, range( 0, 256 )]
102. DWORD NameSize
103. );
104. BOOLEAN RpcWinStationConnect(
105. [in] SERVER\_HANDLE hServer,
106. [out] DWORD \*pResult,
107. [in] DWORD ClientLogonId,
108. [in] DWORD ConnectLogonId,
109. [in] DWORD TargetLogonId,
110. [in,size\_is(PasswordSize)] PWCHAR pPassword,
111. [in, range(0, 1024)]
112. DWORD PasswordSize,
113. [in] BOOLEAN Wait
114. );
115. void Opnum11NotUsedOnWire(void);
116. void Opnum12NotUsedOnWire(void);
117. BOOLEAN RpcWinStationDisconnect(
118. [in] SERVER\_HANDLE hServer,
119. [out] DWORD \*pResult,
120. [in] DWORD LogonId,
121. [in] BOOLEAN bWait
122. );
123. BOOLEAN RpcWinStationReset(
124. [in] SERVER\_HANDLE hServer,
125. [out] DWORD \*pResult,
126. [in] DWORD LogonId,
127. [in] BOOLEAN bWait
128. );
129. BOOLEAN RpcWinStationShutdownSystem(
130. [in] SERVER\_HANDLE hServer,
131. [out] DWORD \*pResult,
132. [in] DWORD ClientLogonId,
133. [in] DWORD ShutdownFlags
134. );
135. BOOLEAN RpcWinStationWaitSystemEvent(
136. [in] SERVER\_HANDLE hServer,
137. [out] DWORD \*pResult,
138. [in] DWORD EventMask,
139. [out] DWORD \*pEventFlags
140. );
141. BOOLEAN RpcWinStationShadow(
142. [in] SERVER\_HANDLE hServer,
143. [out] DWORD \*pResult,
144. [in] DWORD LogonId,
145. [in,unique,size\_is(NameSize)] PWCHAR pTargetServerName,
146. [in, range( 0, 1024)] DWORD NameSize,
147. [in] DWORD TargetLogonId,
148. [in] BYTE HotKeyVk,
149. [in] USHORT HotkeyModifiers
150. );
151. void Opnum18NotUsedOnWire(void);
152. void Opnum19NotUsedOnWire(void);
153. void Opnum20NotUsedOnWire(void);
154. void Opnum21NotUsedOnWire(void);
155. void Opnum22NotUsedOnWire(void);
156. void Opnum23NotUsedOnWire(void);
157. void Opnum24NotUsedOnWire(void);
158. void Opnum25NotUsedOnWire(void);
159. void Opnum26NotUsedOnWire(void);
160. void Opnum27NotUsedOnWire(void);
161. void Opnum28NotUsedOnWire(void);
162. BOOLEAN RpcWinStationBreakPoint(
163. [in] SERVER\_HANDLE hServer,
164. [out] DWORD \*pResult,
165. [in] DWORD LogonId,
166. [in] BOOLEAN KernelFlag
167. );
168. BOOLEAN RpcWinStationReadRegistry(
169. [in] SERVER\_HANDLE hServer,
170. [out] DWORD \*pResult
171. );
172. void Opnum31NotUsedOnWire(void);
173. void Opnum32NotUsedOnWire(void);
174. void Opnum33NotUsedOnWire(void);
175. BOOLEAN OldRpcWinStationEnumerateProcesses(
176. [in] SERVER\_HANDLE hServer,
177. [out] DWORD \*pResult,
178. [out,size\_is(ByteCount)]
179. PBYTE pProcessBuffer,
180. [in, range( 0, 0x8000 )]
181. DWORD ByteCount
182. );
183. void Opnum35NotUsedOnWire(void);
184. BOOLEAN RpcWinStationEnumerateProcesses(
185. [in] SERVER\_HANDLE hServer,
186. [out] DWORD \*pResult,
187. [out,size\_is(ByteCount)]
188. PBYTE pProcessBuffer,
189. [in, range( 0, 0x8000 )]
190. DWORD ByteCount
191. );
192. BOOLEAN RpcWinStationTerminateProcess(
193. [in] SERVER\_HANDLE hServer,
194. [out] DWORD \*pResult,
195. [in] DWORD ProcessId,
196. [in] DWORD ExitCode
197. );
198. void Opnum38NotUsedOnWire(void);
199. void Opnum39NotUsedOnWire(void);
200. void Opnum40NotUsedOnWire(void);
201. void Opnum41NotUsedOnWire(void);
202. void Opnum42NotUsedOnWire(void);
203. BOOLEAN RpcWinStationGetAllProcesses(
204. [in] SERVER\_HANDLE hServer,
205. [out] DWORD \*pResult,
206. [in] ULONG Level,
207. [in, out] BOUNDED\_ULONG \*pNumberOfProcesses,
208. [out, size\_is(,\*pNumberOfProcesses)]
209. PTS\_ALL\_PROCESSES\_INFO \*ppTsAllProcessesInfo
210. );
211. BOOLEAN RpcWinStationGetProcessSid(
212. [in] SERVER\_HANDLE hServer,
213. [in] DWORD dwUniqueProcessId,
214. [in] LARGE\_INTEGER ProcessStartTime,
215. [out] LONG \*pResult,
216. [in,out,unique,size\_is(dwSidSize)]
217. PBYTE pProcessUserSid,
218. [in, range( 0, 1024 )]
219. DWORD dwSidSize,
220. [in,out] DWORD \*pdwSizeNeeded
221. );
222. BOOLEAN RpcWinStationGetTermSrvCountersValue(
223. [in] SERVER\_HANDLE hServer,
224. [out] DWORD \*pResult,
225. [in, range( 0, 0x1000 )]
226. DWORD dwEntries,
227. [in,out, size\_is(dwEntries)]
228. PTS\_COUNTER pCounter
229. );
230. BOOLEAN RpcWinStationReInitializeSecurity(
231. [in] SERVER\_HANDLE hServer,
232. [out] DWORD \*pResult
233. );
234. void Opnum47NotUsedOnWire(void);
235. void Opnum48NotUsedOnWire(void);
236. void Opnum49NotUsedOnWire(void);
237. void Opnum50NotUsedOnWire(void);
238. void Opnum51NotUsedOnWire(void);
239. void Opnum52NotUsedOnWire(void);
240. BOOLEAN RpcWinStationGetLanAdapterName(
241. [in] SERVER\_HANDLE hServer,
242. [out] DWORD \*pResult,
243. [in, range( 0, 0x1000 )]
244. DWORD PdNameSize,
245. [in,size\_is(PdNameSize)] PWCHAR pPdName,
246. [in, range( 0, 1024) ]
247. ULONG LanAdapter,
248. [out] ULONG \*pLength,
249. [out, size\_is(,\*pLength)]
250. PWCHAR \*ppLanAdapter
251. );
252. void Opnum54NotUsedOnWire(void);
253. void Opnum55NotUsedOnWire(void);
254. void Opnum56NotUsedOnWire(void);
255. void Opnum57NotUsedOnWire(void);
256. BOOLEAN RpcWinStationUpdateSettings(
257. [in] SERVER\_HANDLE hServer,
258. [out] DWORD \*pResult,
259. [in] DWORD SettingsClass,
260. [in] DWORD SettingsParameters
261. );
262. BOOLEAN RpcWinStationShadowStop(
263. [in] SERVER\_HANDLE hServer,
264. [out] DWORD \*pResult,
265. [in] DWORD LogonId,
266. [in] BOOLEAN bWait
267. );
268. BOOLEAN RpcWinStationCloseServerEx(
269. [in, out] SERVER\_HANDLE \*phServer,
270. [out] DWORD \*pResult
271. );
272. BOOLEAN RpcWinStationIsHelpAssistantSession(
273. [in] SERVER\_HANDLE hServer,
274. [out] DWORD\* pResult,
275. [in] ULONG SessionId
276. );
277. BOOLEAN RpcWinStationGetMachinePolicy(
278. [in] SERVER\_HANDLE hServer,
279. [in,out,size\_is(bufferSize)]
280. PBYTE pPolicy,
281. [in, range( 0, 0x8000 )]
282. ULONG bufferSize
283. );
284. void Opnum63NotUsedOnWire(void);
285. void Opnum64NotUsedOnWire(void);
286. BOOLEAN RpcWinStationCheckLoopBack(
287. [in] SERVER\_HANDLE hServer,
288. [out] DWORD \*pResult,
289. [in] DWORD ClientLogonId,
290. [in] DWORD TargetLogonId,
291. [in,size\_is(NameSize)]
292. PWCHAR pTargetServerName,
293. [in, range( 0, 1024 )]
294. DWORD NameSize
295. );
296. BOOLEAN RpcConnectCallback(
297. [in] SERVER\_HANDLE hServer,
298. [out] DWORD \*pResult,
299. [in] DWORD TimeOut,
300. [in] ULONG AddressType,
301. [in,size\_is(AddressSize)] PBYTE pAddress,
302. [in, range( 0, 0x1000 )]
303. ULONG AddressSize
304. );
305. void Opnum67NotUsedOnWire(void);
306. void Opnum68NotUsedOnWire(void);
307. void Opnum69NotUsedOnWire(void);
308. BOOLEAN RpcWinStationGetAllProcesses\_NT6(
309. [in] SERVER\_HANDLE hServer,
310. [out] DWORD \*pResult,
311. [in] ULONG Level,
312. [in, out] BOUNDED\_ULONG \*pNumberOfProcesses,
313. [out, size\_is(,\*pNumberOfProcesses)]
314. PTS\_ALL\_PROCESSES\_INFO\_NT6 \*ppTsAllProcessesInfo
315. );
316. void Opnum71NotUsedOnWire(void);
317. void Opnum72NotUsedOnWire(void);
318. void Opnum73NotUsedOnWire(void);
319. void Opnum74NotUsedOnWire(void);
320. BOOLEAN RpcWinStationOpenSessionDirectory(
321. [in] SERVER\_HANDLE hServer,
322. [out] DWORD \*pResult,
323. [in, string, max\_is(64)]
324. PWCHAR pszServerName
325. );
326. }

## Appendix A.4: TSVIPRpc.idl

1. import "ms-dtyp.idl";
2. #include "ms-tsts\_allproc.h"
3. [
4. uuid(53b46b02-c73b-4a3e-8dee-b16b80672fc0),
5. version(1.0),
6. pointer\_default(unique)
7. ]
8. interface TSVIPPublic
9. {
10. HRESULT RpcGetSessionIP(
11. [in] handle\_t hBinding,
12. [in] USHORT Family,
13. [in] DWORD SessionId,
14. [ref, out] TSVIPSession \*ppVIPSession
15. );
16. }

## Appendix A.5: winsta.h

For ease of implementation, the full header is provided.

1. import "ms-dtyp.idl";
2. // [MS-TSTS] specific defines
3. #define WDPREFIX\_LENGTH 12
4. #define STACK\_ADDRESS\_LENGTH 128
5. #define MAX\_BR\_NAME 65
6. #define DIRECTORY\_LENGTH 256
7. #define INITIALPROGRAM\_LENGTH 256
8. #define USERNAME\_LENGTH 20
9. #define DOMAIN\_LENGTH 17
10. #define PASSWORD\_LENGTH 14
11. #define NASISPECIFICNAME\_LENGTH 14
12. #define NASIUSERNAME\_LENGTH 47
13. #define NASIPASSWORD\_LENGTH 24
14. #define NASISESSIONNAME\_LENGTH 16
15. #define NASIFILESERVER\_LENGTH 47
16. #define CLIENTDATANAME\_LENGTH 7
17. #define CLIENTNAME\_LENGTH 20
18. #define CLIENTADDRESS\_LENGTH 30
19. #define IMEFILENAME\_LENGTH 32
20. #define DIRECTORY\_LENGTH 256
21. #define CLIENTLICENSE\_LENGTH 32
22. #define CLIENTMODEM\_LENGTH 40
23. #define CLIENT\_PRODUCT\_ID\_LENGTH 32
24. #define MAX\_COUNTER\_EXTENSIONS 2
25. #define WINSTATIONNAME\_LENGTH 32
26. typedef enum \_SDCLASS {
27. SdNone = 0,
28. SdConsole,
29. SdNetwork,
30. SdAsync,
31. SdOemTransport
32. } SDCLASS;
33. typedef enum \_FLOWCONTROLCLASS {
34. FlowControl\_None,
35. FlowControl\_Hardware,
36. FlowControl\_Software
37. } FLOWCONTROLCLASS;
38. typedef enum \_WINSTATIONSTATECLASS {
39. State\_Active = 0,
40. State\_Connected = 1,
41. State\_ConnectQuery = 2,
42. State\_Shadow = 3,
43. State\_Disconnected = 4,
44. State\_Idle = 5,
45. State\_Listen = 6,
46. State\_Reset = 7,
47. State\_Down = 8,
48. State\_Init = 9
49. } WINSTATIONSTATECLASS;
50. typedef WCHAR NASISPECIFICNAME[ NASISPECIFICNAME\_LENGTH + 1 ];
51. typedef WCHAR NASIUSERNAME[ NASIUSERNAME\_LENGTH + 1 ];
52. typedef WCHAR NASIPASSWORD[ NASIPASSWORD\_LENGTH + 1 ];
53. typedef WCHAR NASISESIONNAME[ NASISESSIONNAME\_LENGTH + 1 ];
54. typedef WCHAR NASIFILESERVER[ NASIFILESERVER\_LENGTH + 1 ];
55. typedef CHAR CLIENTDATANAME[ CLIENTDATANAME\_LENGTH + 1 ];
56. typedef CHAR \* PCLIENTDATANAME;
57. typedef WCHAR WINSTATIONNAME[ WINSTATIONNAME\_LENGTH + 1 ];
58. typedef struct \_TS\_SYSTEMTIME {
59. USHORT wYear;
60. USHORT wMonth;
61. USHORT wDayOfWeek;
62. USHORT wDay;
63. USHORT wHour;
64. USHORT wMinute;
65. USHORT wSecond;
66. USHORT wMilliseconds;
67. } TS\_SYSTEMTIME;
68. typedef struct \_TS\_TIME\_ZONE\_INFORMATION {
69. LONG Bias;
70. WCHAR StandardName[32 ];
71. TS\_SYSTEMTIME StandardDate;
72. LONG StandardBias;
73. WCHAR DaylightName[32 ];
74. TS\_SYSTEMTIME DaylightDate;
75. LONG DaylightBias;
76. } TS\_TIME\_ZONE\_INFORMATION;
77. #ifdef \_WIN64
78. typedef unsigned \_\_int64 ULONG\_PTR;
79. #else
80. typedef ULONG ULONG\_PTR;
81. #endif
82. typedef ULONG PTR\_SIZE\_T;
83. typedef enum {
84. SF\_SERVICES\_SESSION\_POPUP
85. } SESSION\_FILTER;
86. #define PROTOCOL\_CONSOLE 0
87. #define PROTOCOL\_ICA 1
88. #define PROTOCOL\_TSHARE 2
89. #define PROTOCOL\_RDP 2
90. #define PDNAME\_LENGTH 32
91. #define WDNAME\_LENGTH 32
92. #define CDNAME\_LENGTH 32
93. #define DEVICENAME\_LENGTH 128
94. #define MODEMNAME\_LENGTH DEVICENAME\_LENGTH
95. #define CALLBACK\_LENGTH 50
96. #define DLLNAME\_LENGTH 32
97. #define WINSTATIONCOMMENT\_LENGTH 60
98. #define MAX\_LICENSE\_SERVER\_LENGTH 1024
99. #define LOGONID\_CURRENT ((ULONG)-1)
100. #define MAX\_PDCONFIG 10
101. #define TERMSRV\_TOTAL\_SESSIONS 1
102. #define TERMSRV\_DISC\_SESSIONS 2
103. #define TERMSRV\_RECON\_SESSIONS 3
104. #define TERMSRV\_CURRENT\_ACTIVE\_SESSIONS 4
105. #define TERMSRV\_CURRENT\_DISC\_SESSIONS 5
106. #define TERMSRV\_PENDING\_SESSIONS 6
107. #define TERMSRV\_SUCC\_TOTAL\_LOGONS 7
108. #define TERMSRV\_SUCC\_LOCAL\_LOGONS 8
109. #define TERMSRV\_SUCC\_REMOTE\_LOGONS 9
110. #define TERMSRV\_SUCC\_SESSION0\_LOGONS 10
111. #define TERMSRV\_CURRENT\_TERMINATING\_SESSIONS 11
112. #define TERMSRV\_CURRENT\_LOGGEDON\_SESSIONS 12
113. #define NO\_FALLBACK\_DRIVERS 0x0
114. #define FALLBACK\_BESTGUESS 0x1
115. #define FALLBACK\_PCL 0x2
116. #define FALLBACK\_PS 0x3
117. #define FALLBACK\_PCLANDPS 0x4
118. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
119. \* WinStationOpen access values
120. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
121. #define WINSTATION\_QUERY 0x00000001 /\* WinStationQueryInformation() \*/
122. #define WINSTATION\_SET 0x00000002 /\* WinStationSetInformation() \*/
123. #define WINSTATION\_RESET 0x00000004 /\* WinStationReset() \*/
124. #define WINSTATION\_VIRTUAL 0x00000008 /\* read/write direct data \*/
125. #define WINSTATION\_SHADOW 0x00000010 /\* WinStationShadow() \*/
126. #define WINSTATION\_LOGON 0x00000020 /\* logon to WinStation \*/
127. #define WINSTATION\_LOGOFF 0x00000040 /\* WinStationLogoff() \*/
128. #define WINSTATION\_MSG 0x00000080 /\* WinStationMsg() \*/
129. #define WINSTATION\_CONNECT 0x00000100 /\* WinStationConnect() \*/
130. #define WINSTATION\_DISCONNECT 0x00000200 /\* WinStationDisconnect() \*/
131. #define WINSTATION\_GUEST\_ACCESS (WINSTATION\_LOGON)
132. #define WINSTATION\_CURRENT\_GUEST\_ACCESS (WINSTATION\_VIRTUAL | \
133. WINSTATION\_LOGOFF)
134. #define WINSTATION\_USER\_ACCESS (WINSTATION\_GUEST\_ACCESS | \
135. WINSTATION\_QUERY | \
136. WINSTATION\_CONNECT )
137. #define WINSTATION\_CURRENT\_USER\_ACCESS (WINSTATION\_SET | \
138. WINSTATION\_RESET | \
139. WINSTATION\_VIRTUAL | \
140. WINSTATION\_LOGOFF | \
141. WINSTATION\_DISCONNECT)
142. #define WINSTATION\_ALL\_ACCESS ( STANDARD\_RIGHTS\_REQUIRED | \
143. WINSTATION\_QUERY | \
144. WINSTATION\_SET | \
145. WINSTATION\_RESET | \
146. WINSTATION\_VIRTUAL | \
147. WINSTATION\_SHADOW | \
148. WINSTATION\_LOGON | \
149. WINSTATION\_MSG | \
150. WINSTATION\_CONNECT | \
151. WINSTATION\_DISCONNECT)
152. typedef WCHAR PDNAME[ PDNAME\_LENGTH + 1 ];
153. typedef WCHAR \* PPDNAME;
154. /\*------------------------------------------------\*/
155. typedef WCHAR WDNAME[ WDNAME\_LENGTH + 1 ];
156. typedef WCHAR \* PWDNAME;
157. /\*------------------------------------------------\*/
158. typedef WCHAR CDNAME[ CDNAME\_LENGTH + 1 ];
159. typedef WCHAR \* PCDNAME;
160. /\*------------------------------------------------\*/
161. typedef WCHAR DEVICENAME[ DEVICENAME\_LENGTH + 1 ];
162. typedef WCHAR \* PDEVICENAME;
163. /\*------------------------------------------------\*/
164. typedef WCHAR MODEMNAME[ MODEMNAME\_LENGTH + 1 ];
165. typedef WCHAR \* PMODEMNAME;
166. /\*------------------------------------------------\*/
167. typedef WCHAR DLLNAME[ DLLNAME\_LENGTH + 1 ];
168. typedef WCHAR \* PDLLNAME;
169. typedef CHAR DLLNAMEA[ DLLNAME\_LENGTH + 1 ];
170. /\*------------------------------------------------\*/
171. typedef WCHAR WDPREFIX[ WDPREFIX\_LENGTH + 1 ];
172. typedef WCHAR \* PWDPREFIX;
173. /\*
174. \* Stack address structure
175. \*/
176. typedef struct \_CLIENT\_STACK\_ADDRESS {
177. BYTE Address[ STACK\_ADDRESS\_LENGTH ]; // bytes 0,1 family, 2-n address
178. } CLIENT\_STACK\_ADDRESS, \*PCLIENT\_STACK\_ADDRESS;
179. typedef struct \_TS\_TRACE {
180. WCHAR TraceFile[256];
181. BOOLEAN fDebugger;
182. BOOLEAN fTimestamp;
183. ULONG TraceClass;
184. ULONG TraceEnable;
185. WCHAR TraceOption[64];
186. } TS\_TRACE, \* PTS\_TRACE;
187. #define EXTENDED\_USERNAME\_LEN 255
188. #define EXTENDED\_PASSWORD\_LEN 255
189. #define EXTENDED\_DOMAIN\_LEN 255
190. typedef struct \_ExtendedClientCredentials {
191. WCHAR UserName[EXTENDED\_USERNAME\_LEN + 1];
192. WCHAR Password[EXTENDED\_PASSWORD\_LEN + 1];
193. WCHAR Domain[EXTENDED\_DOMAIN\_LEN + 1] ;
194. }ExtendedClientCredentials, \*pExtendedClientCredentials;
195. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
196. \* User Configuration structures
197. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
198. typedef WCHAR APPLICATIONNAME[ MAX\_BR\_NAME ];
199. typedef WCHAR \*PAPPLICATIONNAME;
200. /\*
201. \* Shadow options
202. \*/
203. typedef enum \_SHADOWCLASS {
204. Shadow\_Disable,
205. Shadow\_EnableInputNotify,
206. Shadow\_EnableInputNoNotify,
207. Shadow\_EnableNoInputNotify,
208. Shadow\_EnableNoInputNoNotify,
209. } SHADOWCLASS;
210. /\*
211. \* Callback options
212. \*/
213. typedef enum \_CALLBACKCLASS {
214. Callback\_Disable,
215. Callback\_Roving,
216. Callback\_Fixed,
217. } CALLBACKCLASS;
218. typedef struct \_POLICY\_TS\_MACHINE
219. {
220. ULONG fPolicyDisableClip : 1 ;
221. ULONG fPolicyDisableCam : 1 ;
222. ULONG fPolicyDisableCcm : 1 ;
223. ULONG fPolicyDisableLPT : 1;
224. ULONG fPolicyDisableCpm : 1;
225. ULONG fPolicyPromptForPassword : 1 ;
226. ULONG fPolicyMaxInstanceCount : 1;
227. ULONG fPolicyMinEncryptionLevel : 1 ;
228. ULONG fPolicyFipsEnabled : 1;
229. ULONG fPolicyDisableAutoReconnect : 1;
230. ULONG fPolicyWFProfilePath: 1 ;
231. ULONG fPolicyWFHomeDir: 1 ;
232. ULONG fPolicyWFHomeDirDrive: 1 ;
233. ULONG fPolicyDenyTSConnections : 1;
234. ULONG fPolicyTempFoldersPerSession : 1;
235. ULONG fPolicyDeleteTempFoldersOnExit: 1;
236. ULONG fPolicyColorDepth : 1;
237. ULONG fPolicySessionDirectoryActive : 1;
238. ULONG fPolicySessionDirectoryLocation : 1;
239. ULONG fPolicySessionDirectoryClusterName : 1;
240. ULONG fPolicySessionDirectoryAdditionalParams : 1;
241. ULONG fPolicySessionDirectoryExposeServerIP : 1;
242. ULONG fPolicyPreventLicenseUpgrade : 1;
243. ULONG fPolicySecureLicensing : 1;
244. ULONG fPolicyWritableTSCCPermissionsTAB : 1;
245. ULONG fPolicyDisableCdm : 1;
246. ULONG fPolicyForceClientLptDef : 1;
247. ULONG fPolicyShadow : 1 ;
248. ULONG fPolicyResetBroken : 1 ;
249. ULONG fPolicyReconnectSame : 1 ;
250. ULONG fPolicyMaxSessionTime : 1 ;
251. ULONG fPolicyMaxDisconnectionTime:1;
252. ULONG fPolicyMaxIdleTime : 1 ;
253. ULONG fPolicyInitialProgram : 1 ;
254. ULONG fPolicySingleSessionPerUser : 1;
255. ULONG fPolicyDisableWallpaper : 1;
256. ULONG fPolicyKeepAlive : 1;
257. ULONG fPolicyEnableTimeZoneRedirection : 1;
258. ULONG fPolicyDisableForcibleLogoff : 1;
259. ULONG fPolicyLicensingMode : 1;
260. ULONG fPolicyExplicitLSDiscovery: 1;
261. ULONG fPolicyDisableTerminalServerTooltip:1;
262. ULONG fDisableClip : 1 ;
263. ULONG fDisableCam : 1 ;
264. ULONG fDisableCcm : 1 ;
265. ULONG fDisableLPT : 1;
266. ULONG fDisableCpm : 1;
267. ULONG fPromptForPassword : 1 ;
268. ULONG ColorDepth : 3;
269. ULONG fDenyTSConnections : 1;
270. ULONG fTempFoldersPerSession : 1;
271. ULONG fDeleteTempFoldersOnExit: 1;
272. ULONG fWritableTSCCPermissionsTAB : 1;
273. ULONG fDisableCdm : 1;
274. ULONG fForceClientLptDef : 1;
275. ULONG fResetBroken : 1 ;
276. ULONG fReconnectSame : 1 ;
277. ULONG fSingleSessionPerUser:1;
278. ULONG fDisableWallpaper : 1;
279. ULONG fKeepAliveEnable : 1;
280. ULONG fPreventLicenseUpgrade:1;
281. ULONG fSecureLicensing:1;
282. ULONG fEnableTimeZoneRedirection : 1;
283. ULONG fDisableAutoReconnect : 1;
284. ULONG fDisableForcibleLogoff : 1;
285. ULONG fPolicyEncryptRPCTraffic : 1;
286. ULONG fEncryptRPCTraffic : 1;
287. ULONG fErrorInvalidProfile : 1;
288. ULONG fPolicyFallbackPrintDriver : 1;
289. ULONG FallbackPrintDriverType : 3;
290. ULONG fDisableTerminalServerTooltip : 1;
291. BYTE bSecurityLayer;
292. ULONG fPolicySecurityLayer : 1;
293. BYTE bUserAuthentication;
294. ULONG fPolicyUserAuthentication : 1;
295. ULONG fPolicyTurnOffSingleAppMode : 1;
296. ULONG fTurnOffSingleAppMode : 1;
297. ULONG fDisablePNPPolicyIsEnfored:1;
298. ULONG fDisablePNPPolicyValue:1;
299. ULONG MaxInstanceCount;
300. ULONG LicensingMode;
301. BYTE MinEncryptionLevel;
302. WCHAR WFProfilePath[ DIRECTORY\_LENGTH + 1 ];
303. WCHAR WFHomeDir[ DIRECTORY\_LENGTH + 1 ];
304. WCHAR WFHomeDirDrive[ 4 ];
305. ULONG SessionDirectoryActive;
306. WCHAR SessionDirectoryLocation[DIRECTORY\_LENGTH+1];
307. WCHAR SessionDirectoryClusterName[DIRECTORY\_LENGTH+1];
308. WCHAR SessionDirectoryAdditionalParams[DIRECTORY\_LENGTH+1];
309. ULONG SessionDirectoryExposeServerIP;
310. ULONG KeepAliveInterval;
311. SHADOWCLASS Shadow;
312. ULONG MaxConnectionTime;
313. ULONG MaxDisconnectionTime;
314. ULONG MaxIdleTime;
315. WCHAR WorkDirectory[ DIRECTORY\_LENGTH + 1 ];
316. WCHAR InitialProgram[ INITIALPROGRAM\_LENGTH + 1 ];
317. WCHAR LicenseServers[MAX\_LICENSE\_SERVER\_LENGTH + 1 ];
318. } POLICY\_TS\_MACHINE, \*PPOLICY\_TS\_MACHINE;
319. /\*
320. \* User Configuration data
321. \*/
322. typedef struct \_USERCONFIG {
323. /\* if flag is set inherit parameter from user or client configuration \*/
324. ULONG fInheritAutoLogon : 1;
325. ULONG fInheritResetBroken : 1;
326. ULONG fInheritReconnectSame : 1;
327. ULONG fInheritInitialProgram : 1;
328. ULONG fInheritCallback : 1;
329. ULONG fInheritCallbackNumber : 1;
330. ULONG fInheritShadow : 1;
331. ULONG fInheritMaxSessionTime : 1;
332. ULONG fInheritMaxDisconnectionTime : 1;
333. ULONG fInheritMaxIdleTime : 1;
334. ULONG fInheritAutoClient : 1;
335. ULONG fInheritSecurity : 1;
336. ULONG fPromptForPassword : 1;
337. ULONG fResetBroken : 1;
338. ULONG fReconnectSame : 1;
339. ULONG fLogonDisabled : 1;
340. ULONG fWallPaperDisabled : 1;
341. ULONG fAutoClientDrives : 1;
342. ULONG fAutoClientLpts : 1;
343. ULONG fForceClientLptDef : 1;
344. ULONG fRequireEncryption : 1;
345. ULONG fDisableEncryption : 1;
346. ULONG fUnused1 : 1;
347. ULONG fHomeDirectoryMapRoot : 1;
348. ULONG fUseDefaultGina : 1;
349. ULONG fCursorBlinkDisabled : 1;
350. ULONG fPublishedApp : 1;
351. ULONG fHideTitleBar : 1;
352. ULONG fMaximize : 1;
353. ULONG fDisableCpm : 1;
354. ULONG fDisableCdm : 1;
355. ULONG fDisableCcm : 1;
356. ULONG fDisableLPT : 1;
357. ULONG fDisableClip : 1;
358. ULONG fDisableExe : 1;
359. ULONG fDisableCam : 1;
360. ULONG fDisableAutoReconnect : 1;
361. ULONG ColorDepth : 3;
362. ULONG fInheritColorDepth: 1;
363. ULONG fErrorInvalidProfile : 1;
364. ULONG fPasswordIsScPin: 1;
365. ULONG fDisablePNPRedir:1;
366. WCHAR UserName[ USERNAME\_LENGTH + 1 ];
367. WCHAR Domain[ DOMAIN\_LENGTH + 1 ];
368. WCHAR Password[ PASSWORD\_LENGTH + 1 ];
369. WCHAR WorkDirectory[ DIRECTORY\_LENGTH + 1 ];
370. WCHAR InitialProgram[ INITIALPROGRAM\_LENGTH + 1 ];
371. WCHAR CallbackNumber[ CALLBACK\_LENGTH + 1 ];
372. CALLBACKCLASS Callback;
373. SHADOWCLASS Shadow;
374. ULONG MaxConnectionTime;
375. ULONG MaxDisconnectionTime;
376. ULONG MaxIdleTime;
377. ULONG KeyboardLayout;
378. BYTE MinEncryptionLevel;
379. WCHAR NWLogonServer[ NASIFILESERVER\_LENGTH + 1 ];
380. APPLICATIONNAME PublishedName;
381. WCHAR WFProfilePath[ DIRECTORY\_LENGTH + 1 ];
382. WCHAR WFHomeDir[ DIRECTORY\_LENGTH + 1 ];
383. WCHAR WFHomeDirDrive[ 4 ];
384. } USERCONFIG, \* PUSERCONFIG;
385. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
386. \* PD structures
387. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
388. typedef struct \_PDCONFIG2{
389. PDNAME PdName;
390. SDCLASS SdClass;
391. DLLNAME PdDLL;
392. ULONG PdFlag;
393. ULONG OutBufLength;
394. ULONG OutBufCount;
395. ULONG OutBufDelay;
396. ULONG InteractiveDelay;
397. ULONG PortNumber;
398. ULONG KeepAliveTimeout;
399. } PDCONFIG2, \* PPDCONFIG2;
400. /\*
401. \* PdFlag defines
402. \*/
403. #define PD\_UNUSED 0x00000001
404. #define PD\_RELIABLE 0x00000002
405. #define PD\_FRAME 0x00000004
406. #define PD\_CONNECTION 0x00000008
407. #define PD\_CONSOLE 0x00000010
408. #define PD\_LANA 0x00000020
409. #define PD\_TRANSPORT 0x00000040
410. #define PD\_SINGLE\_INST 0x00000080
411. #define PD\_NOLOW\_WATERMARK 0x00000100
412. /\*------------------------------------------------\*/
413. typedef enum \_RECEIVEFLOWCONTROLCLASS {
414. ReceiveFlowControl\_None,
415. ReceiveFlowControl\_RTS,
416. ReceiveFlowControl\_DTR,
417. } RECEIVEFLOWCONTROLCLASS;
418. typedef enum \_TRANSMITFLOWCONTROLCLASS {
419. TransmitFlowControl\_None,
420. TransmitFlowControl\_CTS,
421. TransmitFlowControl\_DSR,
422. } TRANSMITFLOWCONTROLCLASS;
423. typedef struct \_FLOWCONTROLCONFIG {
424. ULONG fEnableSoftwareTx: 1;
425. ULONG fEnableSoftwareRx: 1;
426. ULONG fEnableDTR: 1;
427. ULONG fEnableRTS: 1;
428. CHAR XonChar;
429. CHAR XoffChar;
430. FLOWCONTROLCLASS Type;
431. RECEIVEFLOWCONTROLCLASS HardwareReceive;
432. TRANSMITFLOWCONTROLCLASS HardwareTransmit;
433. } FLOWCONTROLCONFIG, \* PFLOWCONTROLCONFIG;
434. typedef enum \_ASYNCCONNECTCLASS {
435. Connect\_CTS,
436. Connect\_DSR,
437. Connect\_RI,
438. Connect\_DCD,
439. Connect\_FirstChar,
440. Connect\_Perm,
441. } ASYNCCONNECTCLASS;
442. typedef struct \_CONNECTCONFIG {
443. ASYNCCONNECTCLASS Type;
444. ULONG fEnableBreakDisconnect: 1;
445. } CONNECTCONFIG, \* PCONNECTCONFIG;
446. /\*------------------------------------------------\*/
447. typedef struct \_ASYNCCONFIG {
448. DEVICENAME DeviceName;
449. MODEMNAME ModemName;
450. ULONG BaudRate;
451. ULONG Parity;
452. ULONG StopBits;
453. ULONG ByteSize;
454. ULONG fEnableDsrSensitivity: 1;
455. ULONG fConnectionDriver: 1;
456. FLOWCONTROLCONFIG FlowControl;
457. CONNECTCONFIG Connect;
458. } ASYNCCONFIG, \* PASYNCCONFIG;
459. /\*------------------------------------------------\*/
460. typedef struct \_NETWORKCONFIG {
461. LONG LanAdapter;
462. DEVICENAME NetworkName;
463. ULONG Flags;
464. } NETWORKCONFIG, \* PNETWORKCONFIG;
465. /\*------------------------------------------------\*/
466. typedef struct \_NASICONFIG {
467. NASISPECIFICNAME SpecificName;
468. NASIUSERNAME UserName;
469. NASIPASSWORD PassWord;
470. NASISESIONNAME SessionName;
471. NASIFILESERVER FileServer;
472. BOOLEAN GlobalSession;
473. } NASICONFIG, \* PNASICONFIG;
474. /\*------------------------------------------------\*/
475. typedef struct \_OEMTDCONFIG {
476. LONG Adapter;
477. DEVICENAME DeviceName;
478. ULONG Flags;
479. } OEMTDCONFIG, \* POEMTDCONFIG;
480. /\*------------------------------------------------\*/
481. typedef struct \_PDPARAMS {
482. SDCLASS SdClass;
483. union {
484. NETWORKCONFIG Network;
485. ASYNCCONFIG Async;
486. NASICONFIG Nasi;
487. OEMTDCONFIG OemTd;
488. };
489. } PDPARAMS, \* PPDPARAMS;
490. /\*------------------------------------------------\*/
491. typedef struct \_PDCONFIG {
492. PDCONFIG2 Create;
493. PDPARAMS Params;
494. } PDCONFIG, \* PPDCONFIG;
495. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
496. \* Wd structures
497. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
498. typedef struct \_WDCONFIG {
499. WDNAME WdName;
500. DLLNAME WdDLL;
501. DLLNAME WsxDLL;
502. ULONG WdFlag;
503. ULONG WdInputBufferLength;
504. DLLNAME CfgDLL;
505. WDPREFIX WdPrefix;
506. } WDCONFIG, \* PWDCONFIG;
507. /\*
508. \* WdFlag defines
509. \*/
510. #define WDF\_UNUSED 0x00000001
511. #define WDF\_SHADOW\_SOURCE 0x00000002
512. #define WDF\_SHADOW\_TARGET 0x00000004
513. #define WDF\_OTHER 0x00000008
514. #define WDF\_TSHARE 0x00000010
515. #define WDF\_DYNAMIC\_RECONNECT 0x00000020
516. #define WDF\_USER\_VCIOCTL 0x00000040
517. #define WDF\_SUBDESKTOP 0x00008000
518. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
519. \* Connection Driver structures (CD)
520. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
521. /\*
522. \* connection driver classes
523. \*/
524. typedef enum \_CDCLASS {
525. CdNone,
526. CdModem,
527. CdClass\_Maximum,
528. } CDCLASS;
529. /\*------------------------------------------------\*/
530. typedef struct \_CDCONFIG {
531. CDCLASS CdClass;
532. CDNAME CdName;
533. DLLNAME CdDLL;
534. ULONG CdFlag;
535. } CDCONFIG, \* PCDCONFIG;
536. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
537. \* Window Station structures
538. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
539. typedef struct \_WINSTATIONCREATE {
540. ULONG fEnableWinStation : 1;
541. ULONG MaxInstanceCount;
542. } WINSTATIONCREATE, \* PWINSTATIONCREATE;
543. /\*------------------------------------------------\*/
544. typedef struct \_WINSTATIONCONFIG {
545. WCHAR Comment[ WINSTATIONCOMMENT\_LENGTH + 1 ];
546. USERCONFIG User;
547. char OEMId[4];
548. } WINSTATIONCONFIG, \* PWINSTATIONCONFIG;
549. /\*------------------------------------------------\*/
550. typedef enum \_SessionType {
551. SESSIONTYPE\_UNKNOWN = 0,
552. SESSIONTYPE\_SERVICES,
553. SESSIONTYPE\_LISTENER,
554. SESSIONTYPE\_REGULARDESKTOP,
555. SESSIONTYPE\_ALTERNATESHELL,
556. SESSIONTYPE\_REMOTEAPP,
557. SESSIONTYPE\_MEDIACENTEREXT
558. } SESSIONTYPE;
559. /\*------------------------------------------------\*/
560. #define EXECSRVPIPENAMELEN 48
561. typedef enum \_WINSTATIONINFOCLASS {
562. WinStationCreateData,
563. WinStationConfiguration,
564. WinStationPdParams,
565. WinStationWd,
566. WinStationPd,
567. WinStationPrinter,
568. WinStationClient,
569. WinStationModules,
570. WinStationInformation,
571. WinStationTrace,
572. WinStationBeep,
573. WinStationEncryptionOff,
574. WinStationEncryptionPerm,
575. WinStationNtSecurity,
576. WinStationUserToken,
577. WinStationUnused1,
578. WinStationVideoData,
579. WinStationInitialProgram,
580. WinStationCd,
581. WinStationSystemTrace,
582. WinStationVirtualData,
583. WinStationClientData,
584. WinStationSecureDesktopEnter,
585. WinStationSecureDesktopExit,
586. WinStationLoadBalanceSessionTarget,
587. WinStationLoadIndicator,
588. WinStationShadowInfo,
589. WinStationDigProductId,
590. WinStationLockedState,
591. WinStationRemoteAddress,
592. WinStationIdleTime,
593. WinStationLastReconnectType,
594. WinStationDisallowAutoReconnect,
595. WinStationUnused2,
596. WinStationUnused3,
597. WinStationUnused4,
598. WinStationUnused5,
599. WinStationReconnectedFromId,
600. WinStationEffectsPolicy,
601. WinStationType,
602. WinStationInformationEx
603. } WINSTATIONINFOCLASS;
604. /\*------------------------------------------------\*/
605. typedef struct \_WINSTATIONCLIENTDATA {
606. CLIENTDATANAME DataName;
607. BOOLEAN fUnicodeData;
608. /\* BYTE Data[1]; Variable length data follows \*/
609. } WINSTATIONCLIENTDATA, \* PWINSTATIONCLIENTDATA;
610. /\*------------------------------------------------\*/
611. typedef struct \_WINSTATIONUSERTOKEN {
612. HANDLE ProcessId;
613. HANDLE ThreadId;
614. HANDLE UserToken;
615. } WINSTATIONUSERTOKEN, \* PWINSTATIONUSERTOKEN;
616. /\*------------------------------------------------\*/
617. typedef struct \_WINSTATIONVIDEODATA {
618. USHORT HResolution;
619. USHORT VResolution;
620. USHORT fColorDepth;
621. } WINSTATIONVIDEODATA, \*PWINSTATIONVIDEODATA;
622. /\*----------------------------------------------\*/
623. typedef struct \_WINSTATIONCONFIG2 {
624. WINSTATIONCREATE Create;
625. PDCONFIG Pd[ MAX\_PDCONFIG ];
626. WDCONFIG Wd;
627. CDCONFIG Cd;
628. WINSTATIONCONFIG Config;
629. } WINSTATIONCONFIG2, \* PWINSTATIONCONFIG2;
630. /\*
631. \* WinStation client data structure
632. \*/
633. typedef struct \_WINSTATIONCLIENT {
634. ULONG fTextOnly: 1;
635. ULONG fDisableCtrlAltDel: 1;
636. ULONG fMouse: 1;
637. ULONG fDoubleClickDetect: 1;
638. ULONG fINetClient: 1;
639. ULONG fPromptForPassword : 1;
640. ULONG fMaximizeShell: 1;
641. ULONG fEnableWindowsKey: 1;
642. ULONG fRemoteConsoleAudio: 1;
643. ULONG fPasswordIsScPin: 1;
644. ULONG fNoAudioPlayback: 1;
645. ULONG fUsingSavedCreds: 1;
646. ULONG fRestrictedLogon: 1;
647. WCHAR ClientName[ CLIENTNAME\_LENGTH + 1 ];
648. WCHAR Domain[ DOMAIN\_LENGTH + 1 ];
649. WCHAR UserName[ USERNAME\_LENGTH + 1 ];
650. WCHAR Password[ PASSWORD\_LENGTH + 1 ];
651. WCHAR WorkDirectory[ DIRECTORY\_LENGTH + 1 ];
652. WCHAR InitialProgram[ INITIALPROGRAM\_LENGTH + 1 ];
653. ULONG SerialNumber;
654. BYTE EncryptionLevel;
655. ULONG ClientAddressFamily;
656. WCHAR ClientAddress[ CLIENTADDRESS\_LENGTH + 1 ];
657. USHORT HRes;
658. USHORT VRes;
659. USHORT ColorDepth;
660. USHORT ProtocolType;
661. ULONG KeyboardLayout;
662. ULONG KeyboardType;
663. ULONG KeyboardSubType;
664. ULONG KeyboardFunctionKey;
665. WCHAR imeFileName[ IMEFILENAME\_LENGTH + 1 ];
666. WCHAR ClientDirectory[ DIRECTORY\_LENGTH + 1 ];
667. WCHAR ClientLicense[ CLIENTLICENSE\_LENGTH + 1 ];
668. WCHAR ClientModem[ CLIENTMODEM\_LENGTH + 1 ];
669. ULONG ClientBuildNumber;
670. ULONG ClientHardwareId;
671. USHORT ClientProductId;
672. USHORT OutBufCountHost;
673. USHORT OutBufCountClient;
674. USHORT OutBufLength;
675. WCHAR AudioDriverName[9];
676. TS\_TIME\_ZONE\_INFORMATION ClientTimeZone;
677. ULONG ClientSessionId;
678. WCHAR clientDigProductId[CLIENT\_PRODUCT\_ID\_LENGTH];
679. ULONG PerformanceFlags;
680. ULONG ActiveInputLocale;
681. } WINSTATIONCLIENT, \* PWINSTATIONCLIENT;
682. /\*
683. \* T.Share specific protocol performance counters
684. \*/
685. typedef struct \_TSHARE\_COUNTERS {
686. ULONG Reserved;
687. } TSHARE\_COUNTERS, \* PTSHARE\_COUNTERS;
688. /\*
689. \* WinStation protocol performance counters
690. \*/
691. typedef struct \_PROTOCOLCOUNTERS {
692. ULONG WdBytes;
693. ULONG WdFrames;
694. ULONG WaitForOutBuf;
695. ULONG Frames;
696. ULONG Bytes;
697. ULONG CompressedBytes;
698. ULONG CompressFlushes;
699. ULONG Errors;
700. ULONG Timeouts;
701. ULONG AsyncFramingError;
702. ULONG AsyncOverrunError;
703. ULONG AsyncOverflowError;
704. ULONG AsyncParityError;
705. ULONG TdErrors;
706. USHORT ProtocolType;
707. USHORT Length;
708. union {
709. TSHARE\_COUNTERS TShareCounters;
710. ULONG Reserved[100];
711. } Specific;
712. } PROTOCOLCOUNTERS, \* PPROTOCOLCOUNTERS;
713. /\*
714. \* ThinWire cache statistics
715. \*/
716. typedef struct \_THINWIRECACHE {
717. ULONG CacheReads;
718. ULONG CacheHits;
719. } THINWIRECACHE, \* PTHINWIRECACHE;
720. #define MAX\_THINWIRECACHE 4
721. typedef struct \_RESERVED\_CACHE {
722. THINWIRECACHE ThinWireCache[ MAX\_THINWIRECACHE ];
723. } RESERVED\_CACHE, \* PRESERVED\_CACHE;
724. /\*
725. \* T.Share specific cache statistics
726. \*/
727. typedef struct \_TSHARE\_CACHE {
728. ULONG Reserved;
729. } TSHARE\_CACHE, \* PTSHARE\_CACHE;
730. /\*
731. \* WinStation cache statistics
732. \*/
733. typedef struct CACHE\_STATISTICS {
734. USHORT ProtocolType;
735. USHORT Length;
736. union {
737. RESERVED\_CACHE ReservedCacheStats;
738. TSHARE\_CACHE TShareCacheStats;
739. ULONG Reserved[20];
740. } Specific;
741. } CACHE\_STATISTICS, \* PCACHE\_STATISTICS;
742. /\*
743. \* WinStation protocol status
744. \*/
745. typedef struct \_PROTOCOLSTATUS {
746. PROTOCOLCOUNTERS Output;
747. PROTOCOLCOUNTERS Input;
748. CACHE\_STATISTICS Cache;
749. ULONG AsyncSignal;
750. ULONG AsyncSignalMask;
751. } PROTOCOLSTATUS, \* PPROTOCOLSTATUS;
752. #ifdef \_\_cplusplus
753. typedef struct \_PROTOCOLSTATUSEX : PROTOCOLSTATUS {
754. #else
755. typedef struct \_PROTOCOLSTATUSEX {
756. PROTOCOLSTATUS ;
757. #endif
758. LARGE\_INTEGER Counters[MAX\_COUNTER\_EXTENSIONS];
759. } PROTOCOLSTATUSEX, \*PPROTOCOLSTATUSEX;
760. /\*
761. \* WinStation query information
762. \*/
763. typedef struct \_WINSTATIONINFORMATION {
764. WINSTATIONSTATECLASS ConnectState;
765. WINSTATIONNAME WinStationName;
766. ULONG LogonId;
767. LARGE\_INTEGER ConnectTime;
768. LARGE\_INTEGER DisconnectTime;
769. LARGE\_INTEGER LastInputTime;
770. LARGE\_INTEGER LogonTime;
771. PROTOCOLSTATUS Status;
772. WCHAR Domain[ DOMAIN\_LENGTH + 1 ];
773. WCHAR UserName[USERNAME\_LENGTH + 1];
774. LARGE\_INTEGER CurrentTime;
775. } WINSTATIONINFORMATION, \* PWINSTATIONINFORMATION;
776. /\*
777. \* Load balancing data types
778. \*/
779. typedef enum \_LOADFACTORTYPE {
780. ErrorConstraint,
781. PagedPoolConstraint,
782. NonPagedPoolConstraint,
783. AvailablePagesConstraint,
784. SystemPtesConstraint,
785. CPUConstraint
786. } LOADFACTORTYPE;
787. typedef struct \_WINSTATIONLOADINDICATORDATA {
788. ULONG RemainingSessionCapacity;
789. LOADFACTORTYPE LoadFactor;
790. ULONG TotalSessions;
791. ULONG DisconnectedSessions;
792. LARGE\_INTEGER IdleCPU;
793. LARGE\_INTEGER TotalCPU;
794. ULONG RawSessionCapacity;
795. ULONG reserved[9];
796. } WINSTATIONLOADINDICATORDATA, \* PWINSTATIONLOADINDICATORDATA;
797. /\*
798. \* WinStation shadow states
799. \*/
800. typedef enum \_SHADOWSTATECLASS {
801. State\_NoShadow,
802. State\_Shadowing,
803. State\_Shadowed,
804. } SHADOWSTATECLASS;
805. /\*
806. \* Shadow query/set information
807. \*/
808. typedef struct \_WINSTATIONSHADOW {
809. SHADOWSTATECLASS ShadowState;
810. SHADOWCLASS ShadowClass;
811. ULONG SessionId;
812. ULONG ProtocolType;
813. } WINSTATIONSHADOW, \* PWINSTATIONSHADOW;
814. typedef struct \_WINSTATIONPRODID {
815. WCHAR DigProductId[CLIENT\_PRODUCT\_ID\_LENGTH];
816. WCHAR ClientDigProductId[CLIENT\_PRODUCT\_ID\_LENGTH ];
817. WCHAR OuterMostDigProductId[CLIENT\_PRODUCT\_ID\_LENGTH ];
818. ULONG curentSessionId;
819. ULONG ClientSessionId;
820. ULONG OuterMostSessionId;
821. }WINSTATIONPRODID, \*PWINSTATIONPRODID;
822. typedef struct {
823. unsigned short sin\_family;
824. union {
825. struct {
826. USHORT sin\_port;
827. ULONG in\_addr;
828. UCHAR sin\_zero[8];
829. } ipv4;
830. struct {
831. USHORT sin6\_port;
832. ULONG sin6\_flowinfo;
833. USHORT sin6\_addr[8];
834. ULONG sin6\_scope\_id;
835. } ipv6;
836. };
837. } WINSTATIONREMOTEADDRESS, \*PWINSTATIONREMOTEADDRESS;
838. #define DEFAULT\_POLICY\_ID 1
839. #define PERSEAT\_POLICY\_ID 2
840. #define INTCONN\_POLICY\_ID 3
841. #define PERUSER\_POLICY\_ID 4
842. #define POLICY\_NOT\_CONFIGURED 5
843. #define MAXIMUM\_POLICY\_ID 6
844. /\*------------------------------------------------\*/
845. typedef struct \_BEEPINPUT {
846. ULONG uType;
847. } BEEPINPUT, \* PBEEPINPUT;
848. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
849. \* NWLogon Structure
850. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
851. #define IDTIMEOUT 32000
852. #define IDASYNC 32001
853. #define WSD\_LOGOFF 0x00000001
854. #define WSD\_SHUTDOWN 0x00000002
855. #define WSD\_REBOOT 0x00000004
856. #define WSD\_POWEROFF 0x00000008
857. #define WSD\_FASTREBOOT 0x00000010
858. #define WTS\_CONSOLE\_CONNECT 0x1
859. #define WTS\_CONSOLE\_DISCONNECT 0x2
860. #define WTS\_REMOTE\_CONNECT 0x3
861. #define WTS\_REMOTE\_DISCONNECT 0x4
862. #define WTS\_SESSION\_LOGON 0x5
863. #define WTS\_SESSION\_LOGOFF 0x6
864. #define WTS\_SESSION\_LOCK 0x7
865. #define WTS\_SESSION\_UNLOCK 0x8
866. #define WTS\_SESSION\_REMOTE\_CONTROL 0x9
867. #define CREATE\_MASK(\_\_bit) (1 << (\_\_bit -1) )
868. #define WTS\_CONSOLE\_CONNECT\_MASK CREATE\_MASK( WTS\_CONSOLE\_CONNECT )
869. #define WTS\_CONSOLE\_DISCONNECT\_MASK CREATE\_MASK( WTS\_CONSOLE\_DISCONNECT )
870. #define WTS\_REMOTE\_CONNECT\_MASK CREATE\_MASK( WTS\_REMOTE\_CONNECT )
871. #define WTS\_REMOTE\_DISCONNECT\_MASK CREATE\_MASK( WTS\_REMOTE\_DISCONNECT )
872. #define WTS\_SESSION\_LOGON\_MASK CREATE\_MASK( WTS\_SESSION\_LOGON )
873. #define WTS\_SESSION\_LOGOFF\_MASK CREATE\_MASK( WTS\_SESSION\_LOGOFF )
874. #define WTS\_SESSION\_LOCK\_MASK CREATE\_MASK( WTS\_SESSION\_LOCK )
875. #define WTS\_SESSION\_UNLOCK\_MASK CREATE\_MASK( WTS\_SESSION\_UNLOCK )
876. #define WTS\_SESSION\_REMOTE\_CONTROL\_MASK CREATE\_MASK( WTS\_SESSION\_REMOTE\_CONTROL )
877. #define WTS\_ALL\_NOTIFICATION\_MASK 0xFFFFFFFF
878. typedef struct \_SESSIONID {
879. union {
880. ULONG SessionId;
881. ULONG LogonId;
882. } \_SessionId\_LogonId\_union;
883. WINSTATIONNAME WinStationName;
884. WINSTATIONSTATECLASS State;
885. } SESSIONID, \*PSESSIONID;
886. #define LOGINID SESSIONID
887. #define PLOGINID PSESSIONID
888. #define TS\_USER\_AUTHENTICATION\_NONE 0
889. #define TS\_USER\_AUTHENTICATION\_VIA\_HYBRID 1
890. #define TS\_USER\_AUTHENTICATION\_VIA\_SSL 2
891. #define TS\_USER\_AUTHENTICATION\_DEFAULT TS\_USER\_AUTHENTICATION\_NONE
892. typedef struct \_VARDATA\_WIRE {
893. USHORT Size;
894. USHORT Offset;
895. } VARDATA\_WIRE, \*PVARDATA\_WIRE;
896. typedef struct \_PDPARAMSWIRE {
897. SDCLASS SdClass;
898. VARDATA\_WIRE SdClassSpecific;
899. } PDPARAMSWIRE, \*PPDPARAMSWIRE;
900. typedef struct \_WINSTACONFIGWIRE {
901. WCHAR Comment[61];
902. char OEMId[4];
903. VARDATA\_WIRE UserConfig;
904. VARDATA\_WIRE NewFields;
905. } WINSTACONFIGWIRE, \*PWINSTACONFIGWIRE;
906. #define PRODUCTINFO\_COMPANYNAME\_LENGTH 256
907. #define PRODUCTINFO\_PRODUCTID\_LENGTH 4
908. typedef struct \_WINSTATIONPRODUCTINFO {
909. WCHAR CompanyName[PRODUCTINFO\_COMPANYNAME\_LENGTH];
910. WCHAR ProductID[PRODUCTINFO\_PRODUCTID\_LENGTH];
911. } WINSTATIONPRODUCTINFO, \*PWINSTATIONPRODUCTINFO;
912. #define VALIDATIONINFORMATION\_LICENSE\_LENGTH 16384
913. #define VALIDATIONINFORMATION\_HARDWAREID\_LENGTH 20
914. typedef struct \_WINSTATIONVALIDATIONINFORMATION {
915. WINSTATIONPRODUCTINFO ProductInfo;
916. BYTE License[VALIDATIONINFORMATION\_LICENSE\_LENGTH];
917. ULONG LicenseLength;
918. BYTE HardwareID[VALIDATIONINFORMATION\_HARDWAREID\_LENGTH];
919. ULONG HardwareIDLength;
920. } WINSTATIONVALIDATIONINFORMATION, \*PWINSTATIONVALIDATIONINFORMATION;

## Appendix A.6: tsdef.h

For ease of implementation, the full header file is provided.

1. #ifdef \_\_cplusplus
2. extern "C" {
3. #endif
4. typedef ULONG TNotificationId;
5. #define WTS\_NOTIFY\_NONE ( 0x0 )
6. #define WTS\_NOTIFY\_CREATE ( 0x1 )
7. #define WTS\_NOTIFY\_CONNECT ( 0x2 )
8. #define WTS\_NOTIFY\_DISCONNECT ( 0x4 )
9. #define WTS\_NOTIFY\_LOGON ( 0x8 )
10. #define WTS\_NOTIFY\_LOGOFF ( 0x10 )
11. #define WTS\_NOTIFY\_SHADOW\_START ( 0x20 )
12. #define WTS\_NOTIFY\_SHADOW\_STOP ( 0x40 )
13. #define WTS\_NOTIFY\_TERMINATE ( 0x80 )
14. #define WTS\_NOTIFY\_CONSOLE\_CONNECT ( 0x100 )
15. #define WTS\_NOTIFY\_CONSOLE\_DISCONNECT ( 0x200 )
16. #define WTS\_NOTIFY\_LOCK ( 0x400 )
17. #define WTS\_NOTIFY\_UNLOCK ( 0x800 )
18. #define WTS\_NOTIFY\_ALL ( 0xffffffff )
19. typedef enum \_WINSTATIONUPDATECFGCLASS {
20. WINSTACFG\_LEGACY,
21. WINSTACFG\_SESSDIR
22. } WINSTATIONUPDATECFGCLASS;
23. typedef struct \_SESSION\_CHANGE
24. {
25. LONG SessionId;
26. TNotificationId NotificationId;
27. } SESSION\_CHANGE;
28. typedef struct \_SESSION\_CHANGE \*PSESSION\_CHANGE;
29. #ifndef \_TS\_TIME\_ZONE\_INFORMATION\_
30. #define \_TS\_TIME\_ZONE\_INFORMATION\_
31. typedef struct \_TS\_SYSTEMTIME {
32. USHORT wYear;
33. USHORT wMonth;
34. USHORT wDayOfWeek;
35. USHORT wDay;
36. USHORT wHour;
37. USHORT wMinute;
38. USHORT wSecond;
39. USHORT wMilliseconds;
40. } TS\_SYSTEMTIME;
41. typedef struct \_TS\_TIME\_ZONE\_INFORMATION {
42. LONG Bias;
43. WCHAR StandardName[ 32 ];
44. TS\_SYSTEMTIME StandardDate;
45. LONG StandardBias;
46. WCHAR DaylightName[ 32 ];
47. TS\_SYSTEMTIME DaylightDate;
48. LONG DaylightBias;
49. } TS\_TIME\_ZONE\_INFORMATION;
50. #endif //\_TS\_TIME\_ZONE\_INFORMATION\_
51. typedef enum \_SESSION\_FILTER {
52. SF\_SERVICES\_SESSION\_POPUP
53. } SESSION\_FILTER;
54. typedef struct \_CLIENT\_ID {
55. HANDLE UniqueProcess;
56. HANDLE UniqueThread;
57. } CLIENT\_ID;
58. typedef struct \_SYSTEM\_THREAD\_INFORMATION {
59. LARGE\_INTEGER KernelTime;
60. LARGE\_INTEGER UserTime;
61. LARGE\_INTEGER CreateTime;
62. ULONG WaitTime;
63. PVOID StartAddress;
64. CLIENT\_ID ClientId;
65. LONG Priority;
66. LONG BasePriority;
67. ULONG ContextSwitches;
68. ULONG ThreadState;
69. ULONG WaitReason;
70. } SYSTEM\_THREAD\_INFORMATION, \*PSYSTEM\_THREAD\_INFORMATION;
71. #ifdef \_\_cplusplus
72. }
73. #endif

## Appendix A.7: allproc.h

For ease of implementation, the full header file is provided.

1. #ifndef TS\_ALLPROC\_ALREADY\_SET
2. #define TS\_ALLPROC\_ALREADY\_SET
3. #ifdef \_\_midl
4. cpp\_quote( "#define TS\_PROCESS\_INFO\_MAGIC\_NT4 0x23495452" )
5. #else
6. #define TS\_PROCESS\_INFO\_MAGIC\_NT4 0x23495452
7. #endif
8. typedef struct \_TS\_PROCESS\_INFORMATION\_NT4 {
9. ULONG MagicNumber;
10. ULONG LogonId;
11. PVOID ProcessSid;
12. ULONG Pad;
13. } TS\_PROCESS\_INFORMATION\_NT4, \* PTS\_PROCESS\_INFORMATION\_NT4;
14. // sizes of TS4.0 structures (size has changed in Windows 2000)
15. #ifdef \_\_midl
16. cpp\_quote( "#define SIZEOF\_TS4\_SYSTEM\_THREAD\_INFORMATION 64" )
17. cpp\_quote( "#define SIZEOF\_TS4\_SYSTEM\_PROCESS\_INFORMATION 136" )
18. #else
19. #define SIZEOF\_TS4\_SYSTEM\_THREAD\_INFORMATION 64
20. #define SIZEOF\_TS4\_SYSTEM\_PROCESS\_INFORMATION 136
21. #endif
22. #ifdef \_\_midl
23. cpp\_quote( "#define GAP\_LEVEL\_BASIC 0" )
24. #else
25. #define GAP\_LEVEL\_BASIC 0
26. #endif
27. typedef struct \_TS\_UNICODE\_STRING {
28. USHORT Length;
29. USHORT MaximumLength;
30. #ifdef \_\_midl
31. [size\_is(MaximumLength),length\_is(Length)]PWSTR Buffer;
32. #else
33. PWSTR Buffer;
34. #endif
35. } TS\_UNICODE\_STRING;
36. typedef struct \_TS\_SYS\_PROCESS\_INFORMATION {
37. ULONG NextEntryOffset;
38. ULONG NumberOfThreads;
39. LARGE\_INTEGER SpareLi1;
40. LARGE\_INTEGER SpareLi2;
41. LARGE\_INTEGER SpareLi3;
42. LARGE\_INTEGER CreateTime;
43. LARGE\_INTEGER UserTime;
44. LARGE\_INTEGER KernelTime;
45. TS\_UNICODE\_STRING ImageName;
46. LONG BasePriority; // KPRIORITY in ntexapi.h
47. DWORD UniqueProcessId; // HANDLE in ntexapi.h
48. DWORD InheritedFromUniqueProcessId; // HANDLE in ntexapi.h
49. ULONG HandleCount;
50. ULONG SessionId;
51. ULONG SpareUl3;
52. SIZE\_T PeakVirtualSize;
53. SIZE\_T VirtualSize;
54. ULONG PageFaultCount;
55. ULONG PeakWorkingSetSize;
56. ULONG WorkingSetSize;
57. SIZE\_T QuotaPeakPagedPoolUsage;
58. SIZE\_T QuotaPagedPoolUsage;
59. SIZE\_T QuotaPeakNonPagedPoolUsage;
60. SIZE\_T QuotaNonPagedPoolUsage;
61. SIZE\_T PagefileUsage;
62. SIZE\_T PeakPagefileUsage;
63. SIZE\_T PrivatePageCount;
64. }
65. TS\_SYS\_PROCESS\_INFORMATION, \*PTS\_SYS\_PROCESS\_INFORMATION;
66. typedef struct \_TS\_ALL\_PROCESSES\_INFO {
67. PTS\_SYS\_PROCESS\_INFORMATION pTsProcessInfo;
68. DWORD SizeOfSid;
69. #ifdef \_\_midl
70. [size\_is(SizeOfSid)] PBYTE pSid;
71. #else
72. PBYTE pSid;
73. #endif
74. }
75. TS\_ALL\_PROCESSES\_INFO, \*PTS\_ALL\_PROCESSES\_INFO;
76. //=======================================================================
77. // The following structures are defined for taking care of interface
78. // change in Whistler.
79. typedef struct \_NT6\_TS\_UNICODE\_STRING {
80. USHORT Length;
81. USHORT MaximumLength;
82. #ifdef \_\_midl
83. [size\_is(MaximumLength / 2),length\_is(Length / 2)]PWSTR Buffer;
84. #else
85. PWSTR Buffer;
86. #endif
87. } NT6\_TS\_UNICODE\_STRING;
88. typedef struct \_TS\_SYS\_PROCESS\_INFORMATION\_NT6 {
89. ULONG NextEntryOffset;
90. ULONG NumberOfThreads;
91. LARGE\_INTEGER SpareLi1;
92. LARGE\_INTEGER SpareLi2;
93. LARGE\_INTEGER SpareLi3;
94. LARGE\_INTEGER CreateTime;
95. LARGE\_INTEGER UserTime;
96. LARGE\_INTEGER KernelTime;
97. NT6\_TS\_UNICODE\_STRING ImageName;
98. LONG BasePriority; // KPRIORITY in ntexapi.h
99. DWORD UniqueProcessId; // HANDLE in ntexapi.h
100. DWORD InheritedFromUniqueProcessId; // HANDLE in ntexapi.h
101. ULONG HandleCount;
102. ULONG SessionId;
103. ULONG SpareUl3;
104. SIZE\_T PeakVirtualSize;
105. SIZE\_T VirtualSize;
106. ULONG PageFaultCount;
107. ULONG PeakWorkingSetSize;
108. ULONG WorkingSetSize;
109. SIZE\_T QuotaPeakPagedPoolUsage;
110. SIZE\_T QuotaPagedPoolUsage;
111. SIZE\_T QuotaPeakNonPagedPoolUsage;
112. SIZE\_T QuotaNonPagedPoolUsage;
113. SIZE\_T PagefileUsage;
114. SIZE\_T PeakPagefileUsage;
115. SIZE\_T PrivatePageCount;
116. }
117. TS\_SYS\_PROCESS\_INFORMATION\_NT6, \*PTS\_SYS\_PROCESS\_INFORMATION\_NT6;
118. typedef struct \_TS\_ALL\_PROCESSES\_INFO\_NT6 {
119. PTS\_SYS\_PROCESS\_INFORMATION\_NT6 pTsProcessInfo;
120. DWORD SizeOfSid;
121. #ifdef \_\_midl
122. [size\_is(SizeOfSid)] PBYTE pSid;
123. #else
124. PBYTE pSid;
125. #endif
126. }
127. TS\_ALL\_PROCESSES\_INFO\_NT6, \*PTS\_ALL\_PROCESSES\_INFO\_NT6;
128. //=========================================================================================
129. //
130. // TermSrv Counter Header
131. //
132. typedef struct \_TS\_COUNTER\_HEADER {
133. DWORD dwCounterID; // identifies counter
134. boolean bResult; // result of operation performed on counter
135. } TS\_COUNTER\_HEADER, \*PTS\_COUNTER\_HEADER;
136. typedef struct \_TS\_COUNTER {
137. TS\_COUNTER\_HEADER counterHead;
138. DWORD dwValue; // returned value
139. LARGE\_INTEGER startTime; // start time for counter
140. } TS\_COUNTER, \*PTS\_COUNTER;
141. #endif // TS\_ALLPROC\_ALREADY\_SET
142. #define TSVIP\_MAX\_ADAPTER\_ADDRESS\_LENGTH 16
143. typedef struct \_TSVIP\_SOCKADDR {
144. #ifdef \_\_midl
145. union switch (unsigned short sin\_family) u
146. {
147. case 2: // AF\_INET
148. struct {
149. USHORT sin\_port;
150. ULONG in\_addr;
151. UCHAR sin\_zero[8];
152. } ipv4;
153. case 23: // AF\_INET6
154. struct {
155. USHORT sin6\_port;
156. ULONG sin6\_flowinfo;
157. USHORT sin6\_addr[8];
158. ULONG sin6\_scope\_id;
159. } ipv6;
160. };
161. #else
162. USHORT sin\_family;
163. union
164. {
165. struct {
166. USHORT sin\_port;
167. ULONG in\_addr;
168. UCHAR sin\_zero[8];
169. } ipv4;
170. struct {
171. USHORT sin6\_port;
172. ULONG sin6\_flowinfo;
173. USHORT sin6\_addr[8];
174. ULONG sin6\_scope\_id;
175. } ipv6;
176. } u;
177. #endif
178. } TSVIP\_SOCKADDR,
179. \*PTSVIP\_SOCKADDR;
180. typedef struct \_TSVIPAddress {
181. DWORD dwVersion; //Structure version
182. TSVIP\_SOCKADDR IPAddress; //IPv4 is in network byte order.
183. ULONG PrefixOrSubnetMask; //IPv4 is a mask in network byte order,
184. #ifdef \_\_midl //IPv6 is prefix length.
185. [range(0, TSVIP\_MAX\_ADAPTER\_ADDRESS\_LENGTH)]
186. UINT PhysicalAddressLength;
187. [length\_is(PhysicalAddressLength)]
188. BYTE PhysicalAddress[TSVIP\_MAX\_ADAPTER\_ADDRESS\_LENGTH];
189. #else
190. UINT PhysicalAddressLength;
191. BYTE PhysicalAddress[TSVIP\_MAX\_ADAPTER\_ADDRESS\_LENGTH];
192. #endif
193. ULONG LeaseExpires;
194. ULONG T1;
195. ULONG T2;
196. } TSVIPAddress,
197. \*PTSVIPAddress;
198. typedef struct \_TSVIPSession {
199. DWORD dwVersion; //Structure version
200. DWORD SessionId; //Session ID
201. TSVIPAddress SessionIP; //IPAddress assign to session
202. } TSVIPSession,
203. \*PTSVIPSession;
204. //NBD end

## Appendix A.8: SessEnvRpc.idl

1. //
2. // Interface for accessing SessionEnv functionality.
3. // Public interface
4. //
5. import "ms-dtyp.idl";
6. typedef enum
7. {
8. SHADOW\_CONTROL\_REQUEST\_VIEW = 0,
9. SHADOW\_CONTROL\_REQUEST\_TAKECONTROL,
10. SHADOW\_CONTROL\_REQUEST\_Count
11. } SHADOW\_CONTROL\_REQUEST;
12. typedef enum
13. {
14. SHADOW\_PERMISSION\_REQUEST\_SILENT = 0,
15. SHADOW\_PERMISSION\_REQUEST\_REQUESTPERMISSION,
16. SHADOW\_PERMISSION\_REQUEST\_Count
17. } SHADOW\_PERMISSION\_REQUEST;
18. typedef enum
19. {
20. SHADOW\_REQUEST\_RESPONSE\_ALLOW = 0,
21. SHADOW\_REQUEST\_RESPONSE\_DECLINE,
22. SHADOW\_REQUEST\_RESPONSE\_POLICY\_PERMISSION\_REQUIRED,
23. SHADOW\_REQUEST\_RESPONSE\_POLICY\_DISABLED,
24. SHADOW\_REQUEST\_RESPONSE\_POLICY\_VIEW\_ONLY,
25. SHADOW\_REQUEST\_RESPONSE\_POLICY\_VIEW\_ONLY\_PERMISSION\_REQUIRED,
26. SHADOW\_REQUEST\_RESPONSE\_SESSION\_ALREADY\_CONTROLLED
27. } SHADOW\_REQUEST\_RESPONSE;
28. [
29. uuid(1257B580-CE2F-4109-82D6-A9459D0BF6BC),
30. version(1.0),
31. pointer\_default(unique)
32. ]
33. interface SessEnvPublicRpc
34. {
35. cpp\_quote( "#ifndef \_SESSENVPUBLICRPC\_ENDPOINT\_DEF" )
36. cpp\_quote( "#define \_SESSENVPUBLICRPC\_ENDPOINT\_DEF" )
37. cpp\_quote( "static LPCWSTR SESSENVPUBLICRPC\_ENDPOINT = L\"\\\\pipe\\\\SessEnvPublicRpc\"; ")
38. cpp\_quote( "static LPCWSTR SESSENVPUBLICRPC\_PROTOCOL\_SEQUENCE = L\"ncacn\_np\"; ")
39. cpp\_quote( "#endif // !\_SESSENVPUBLICRPC\_ENDPOINT\_DEF" )
40. // Purpose:
41. // Create shadow invitation
42. //
43. // Parameters:
44. //
45. // IN hBinding
46. // IN TargetSessionId - session ID.
47. // IN eRequestControl -request control or view only
48. // IN eRequestPermission -request permission or silent
49. // OUT pePermission -shadow request or permission request response
50. // OUT pszInvitation -collab API invitation string
51. // IN cchInvitation - size of pszInvitation in WCHARs
52. //
53. HRESULT RpcShadow2(
54. [in] handle\_t hBinding,
55. [in] ULONG TargetSessionId,
56. [in] SHADOW\_CONTROL\_REQUEST eRequestControl,
57. [in] SHADOW\_PERMISSION\_REQUEST eRequestPermission,
58. [out] SHADOW\_REQUEST\_RESPONSE \* pePermission,
59. [out, string, size\_is(cchInvitation)] LPWSTR pszInvitation,
60. [in,range(1, 8192)] ULONG cchInvitation
61. );
62. }

# Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include updates to those products.

* Windows NT operating system
* Windows 2000 operating system
* Windows XP operating system
* Windows Server 2003 operating system
* Windows Vista operating system
* Windows Server 2008 operating system
* Windows 7 operating system
* Windows Server 2008 R2 operating system
* Windows 8 operating system
* Windows Server 2012 operating system
* Windows 8.1 operating system
* Windows Server 2012 R2 operating system
* Windows 10 operating system
* Windows Server 2016 operating system
* Windows Server operating system

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates 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 1.3](#Appendix_A_Target_1): Virtual machines that are running Windows NT and Windows 2000 operating systems are not supported and are not visible to **VM Host Agent** callers.

[<2> Section 1.3](#Appendix_A_Target_2): LSM Session, LSM Notification, LSM Enumeration, TermService, and TermService Listener are not available in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

Legacy is available in Windows (except Windows NT). VM Host Agent Session, VM Host Agent Notification, and VM Host Agent Enumeration are available in Windows Server 2008 R2 operating system.

[<3> Section 1.8](#Appendix_A_Target_3): Windows uses only the values specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90).

[<4> Section 1.9](#Appendix_A_Target_4): Does not exist in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<5> Section 1.9](#Appendix_A_Target_5): Supported in Windows 7, Windows 8, Windows 8.1, and Windows 10.

[<6> Section 1.9](#Appendix_A_Target_6): Does not exist in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<7> Section 1.9](#Appendix_A_Target_7): Supported in Windows 7, Windows 8, Windows 8.1, and Windows 10.

[<8> Section 1.9](#Appendix_A_Target_8): Does not exist in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<9> Section 1.9](#Appendix_A_Target_9): Supported in Windows 7, Windows 8, Windows 8.1, and Windows 10.

[<10> Section 1.9](#Appendix_A_Target_10): Does not exist in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<11> Section 1.9](#Appendix_A_Target_11): Does not exist in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<12> Section 2.2.1.8](#Appendix_A_Target_12): Not supported on Windows NT, Windows 2000, Windows XP, and Windows Server 2003. In Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, Windows Server 2012, Windows 8.1, and Windows Server 2012 R2, **WinStationConnectState** is not defined; instead, the following two enum values are used:

| Value | Description |
| --- | --- |
| WinStationReconnectedFromId | In case of reconnected sessions, this will return the session ID of the temp session from which it was reconnected, or -1 if there is no temp session. |
| WinStationEffectsPolicy | SKU-Differentiation policy for the DWM-over-Terminal Services. |

[<13> Section 2.2.1.8](#Appendix_A_Target_13): WinStationCreateData is not supported on Windows NT and Windows 2000.

[<14> Section 2.2.1.8](#Appendix_A_Target_14): WinStationPrinter is not supported on Windows NT and Windows 2000.

[<15> Section 2.2.1.8](#Appendix_A_Target_15): WinStationTrace is not supported on Windows NT and Windows 2000.

[<16> Section 2.2.1.8](#Appendix_A_Target_16): WinStationBeep not supported on Windows NT and Windows 2000.

[<17> Section 2.2.1.8](#Appendix_A_Target_17): WinStationEncryptionOff is not supported on Windows NT and Windows 2000.

[<18> Section 2.2.1.8](#Appendix_A_Target_18): WinStationEncryptionPerm is not supported on Windows NT and Windows 2000.

[<19> Section 2.2.1.8](#Appendix_A_Target_19): Used by Winlogon.

[<20> Section 2.2.1.8](#Appendix_A_Target_20): WinStationNtSecurity is supported only on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<21> Section 2.2.1.8](#Appendix_A_Target_21): WinStationVideoData is supported only on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<22> Section 2.2.1.8](#Appendix_A_Target_22): WinStationInitialProgram is supported only on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<23> Section 2.2.1.8](#Appendix_A_Target_23): WinStationCd is supported only on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<24> Section 2.2.1.8](#Appendix_A_Target_24): WinStationSystemTrace is supported only on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<25> Section 2.2.1.8](#Appendix_A_Target_25): WinStationClientData is supported only on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<26> Section 2.2.1.8](#Appendix_A_Target_26): WinStationSecureDesktopEnter is supported only on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<27> Section 2.2.1.8](#Appendix_A_Target_27): WinStationSecureDesktopExit is supported only on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<28> Section 2.2.1.8](#Appendix_A_Target_28): WinStationLoadBalanceSessionTarget is supported only on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<29> Section 2.2.1.8](#Appendix_A_Target_29): WinStationShadowInfo is supported only on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<30> Section 2.2.1.8](#Appendix_A_Target_30): WinStationDigProductId is supported only on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<31> Section 2.2.1.8](#Appendix_A_Target_31): Used by Winlogon.

[<32> Section 2.2.1.8](#Appendix_A_Target_32): WinStationIdleTime is supported only on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<33> Section 2.2.1.8](#Appendix_A_Target_33): WinStationLastReconnectType is supported only on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<34> Section 2.2.1.8](#Appendix_A_Target_34): WinStationDisallowAutoReconnect is supported only on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<35> Section 2.2.1.8](#Appendix_A_Target_35): WinStationType is not supported on Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, or Windows Server 2008. For [**Terminal Services**](#gt_ffff3f01-8c21-44d3-bbda-0062a1fbda4b), the supported session types are:

| Value | Meaning |
| --- | --- |
| TERMINAL\_TYPE\_SERVICE  88f5767d-d13f-404d-a348-8b8e030294a9 | The GUID for the server session. |
| TERMINAL\_TYPE\_REGULAR\_DESKTOP  0f0a4bf8-8362-435d-938c-222a518a8b78 | The GUID for the regular Terminal Services session. |
| TERMINAL\_TYPE\_RDP\_REMOTEAP  eddcc3ce-6e7e-4f4b-8439-3d9ad4c9440f | The GUID for the Terminal Services Remote Applications session. |
| TERMINAL\_TYPE\_MCE  8dc86f1d-9969-4379-91c1-06fe1dc60575 | The GUID for the Windows Media Center Edition session. For more information, see [[MSFT-WINMCE]](https://go.microsoft.com/fwlink/?LinkId=208408). |

[<36> Section 2.2.1.8](#Appendix_A_Target_36): WinStationInformationEx is not supported on Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, or Windows Server 2008.

[<37> Section 2.2.1.9](#Appendix_A_Target_37): State\_Reset is not supported on Windows NT or Windows 2000.

[<38> Section 2.2.1.10](#Appendix_A_Target_38): For Terminal Services, the binaries indicated are tdnetb.dll, tdspx.dll, tdftp.dll tdipx.dll.

[<39> Section 2.2.1.10](#Appendix_A_Target_39): For Terminal Services, the binary indicated is tdasync.dll.

[<40> Section 2.2.1.10](#Appendix_A_Target_40): Not used by Terminal Services.

[<41> Section 2.2.1.18](#Appendix_A_Target_41): In Windows the default or standard shell is explorer.exe. It is specified in ‘HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell.

[<42> Section 2.2.2.3](#Appendix_A_Target_42): For Windows this will be ULONG with 1 indicating Desktop Window Manager (DWM) is enabled in the Terminal Services session, disabled otherwise. For more information about DWM, see [[MSDN-DWM]](https://go.microsoft.com/fwlink/?LinkId=122186).

[<43> Section 2.2.2.3](#Appendix_A_Target_43): QUERY\_SESSION\_DATA\_LICENSE is not supported on Windows NT, Windows 2000, or Windows Server 2003.

[<44> Section 2.2.2.3](#Appendix_A_Target_44): QUERY\_SESSION\_DATA\_DEVICEID is not supported on Windows NT, Windows 2000, or Windows Server 2003.

[<45> Section 2.2.2.3](#Appendix_A_Target_45): QUERY\_SESSION\_DATA\_LICENSE\_VALIDATION is not supported on Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, or Windows Server 2008.

[<46> Section 2.2.2.4.1.2](#Appendix_A_Target_46): SessionType is not supported on Windows NT, Windows 2000, Windows XP, or Windows Server 2003. For Terminal Services, the supported session types are as follows.

| Value | Meaning |
| --- | --- |
| TERMINAL\_TYPE\_SERVICE  88f5767d-d13f-404d-a348-8b8e030294a9 | [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) for Service session. |
| TERMINAL\_TYPE\_REGULAR\_DESKTOP  0f0a4bf8-8362-435d-938c-222a518a8b78 | GUID for regular Terminal Services session. |
| TERMINAL\_TYPE\_RDP\_REMOTEAP  eddcc3ce-6e7e-4f4b-8439-3d9ad4c9440f | GUID for Terminal Services Remote Applications session. |
| TERMINAL\_TYPE\_MCE  8dc86f1d-9969-4379-91c1-06fe1dc60575 | GUID for Windows Media Center Edition session. For more information, see [MSFT-WINMCE]. |

[<47> Section 2.2.2.6](#Appendix_A_Target_47): PEXECENVDATA is not supported on Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, or Windows Server 2008.

[<48> Section 2.2.2.6.1](#Appendix_A_Target_48): ExecEnvData is not supported on Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, or Windows Server 2008.

[<49> Section 2.2.2.6.1](#Appendix_A_Target_49): ExecEnvEnum\_Level1 is not supported on Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, or Windows Server 2008.

[<50> Section 2.2.2.6.1](#Appendix_A_Target_50): ExecEnvEnum\_Level2 is not supported on Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, or Windows Server 2008.

[<51> Section 2.2.2.6.1.1](#Appendix_A_Target_51): ExecEnvId is not supported on Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, or Windows Server 2008.

[<52> Section 2.2.2.6.1.2](#Appendix_A_Target_52): ExecEnvId is not supported on Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, or Windows Server 2008.

[<53> Section 2.2.2.7](#Appendix_A_Target_53): PEXECENVDATAEX is not supported on Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows Server 2008 R2, or Windows 7.

[<54> Section 2.2.2.7.1](#Appendix_A_Target_54): ExecEnvDataEx is not supported on Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows Server 2008 R2, or Windows 7.

[<55> Section 2.2.2.7.1](#Appendix_A_Target_55): ExecEnvEnum\_Level1 is not supported on Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows Server 2008 R2, or Windows 7.

[<56> Section 2.2.2.17.1](#Appendix_A_Target_56): TERMSRV\_PENDING\_SESSIONS is not supported on Windows NT, Windows 2000, Windows XP, or Windows Server 2003.

[<57> Section 2.2.2.17.1](#Appendix_A_Target_57): TERMSRV\_SUCC\_TOTAL\_LOGONS is not supported in Windows NT, Windows 2000, or Windows XP.

[<58> Section 2.2.2.17.1](#Appendix_A_Target_58): TERMSRV\_SUCC\_LOCAL\_LOGONS is not supported in Windows NT, Windows 2000, or Windows XP.

[<59> Section 2.2.2.17.1](#Appendix_A_Target_59): TERMSRV\_SUCC\_REMOTE\_LOGONS is not supported in Windows NT, Windows 2000, or Windows XP.

[<60> Section 2.2.2.17.1](#Appendix_A_Target_60): TERMSRV\_SUCC\_SESSION0\_LOGONS is not supported in Windows NT, Windows 2000, or Windows XP.

[<61> Section 2.2.2.17.1](#Appendix_A_Target_61): TERMSRV\_CURRENT\_TERMINATING\_SESSIONS is not supported in Windows NT, Windows 2000, Windows XP, or Windows Server 2003.

[<62> Section 2.2.2.17.1](#Appendix_A_Target_62): TERMSRV\_CURRENT\_LOGGEDON\_SESSIONS is not supported in Windows NT, Windows 2000, Windows XP, or Windows Server 2003.

[<63> Section 2.2.2.18](#Appendix_A_Target_63): Not used by Terminal Services.

[<64> Section 2.2.2.18](#Appendix_A_Target_64): Not used by Terminal Services.

[<65> Section 2.2.2.18](#Appendix_A_Target_65): fLogonDisabled is only supported on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<66> Section 2.2.2.18](#Appendix_A_Target_66): fUseDefaultGina is supported only on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<67> Section 2.2.2.18](#Appendix_A_Target_67): fCursorBlinkDisabled is not supported on Windows NT or Windows 2000.

[<68> Section 2.2.2.18](#Appendix_A_Target_68): fDisableAutoReconnect is not supported on Windows NT or Windows 2000.

[<69> Section 2.2.2.18](#Appendix_A_Target_69): ColorDepth is not supported on Windows NT and Windows 2000.

The supported values 1, 2, 3, 4, and 5 are translated, respectively, as the following number of colors supported: 256 (8 bpp), 32,768 (15 bpp), 65,536 (16 bpp), 16 million (24 bpp), and 16 million with transparency (32 bpp). A ColorDepth value of 5 indicates 32-bit color; this is not supported on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<70> Section 2.2.2.18](#Appendix_A_Target_70): fInheritColorDepth is not supported on Windows NT or Windows 2000.

[<71> Section 2.2.2.18](#Appendix_A_Target_71): fErrorInvalidProfile is not supported on Windows NT, Windows 2000, or Windows XP.

[<72> Section 2.2.2.18](#Appendix_A_Target_72): fPasswordIsScPin is not supported on Windows NT, Windows 2000, or Windows XP.

[<73> Section 2.2.2.18](#Appendix_A_Target_73): In Windows, the default program is explorer.exe.

[<74> Section 2.2.2.18](#Appendix_A_Target_74): Not used by Terminal Services.

[<75> Section 2.2.2.18](#Appendix_A_Target_75): Not used by Terminal Services.

[<76> Section 2.2.2.18](#Appendix_A_Target_76): The **NWLogonServer** value is not used by Terminal Services, although it can be set and retrieved as a part of user and session configuration data.

[<77> Section 2.2.2.18.1](#Appendix_A_Target_77): Callback is not used by Terminal Services, although callback information can be set and retrieved as a part of user and session configuration data.

[<78> Section 2.2.2.19](#Appendix_A_Target_78): fRemoteConsoleAudio is not supported on Windows NT or Windows 2000.

[<79> Section 2.2.2.19](#Appendix_A_Target_79): fPasswordIsScPin is not supported on Windows NT, Windows 2000, or Windows XP.

[<80> Section 2.2.2.19](#Appendix_A_Target_80): fNoAudioPlayback is not supported on Windows NT, Windows 2000, Windows XP, or Windows Server 2003.

[<81> Section 2.2.2.19](#Appendix_A_Target_81): fRestrictedLogon is not supported in Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2, Windows 8, or Windows Server 2012.

[<82> Section 2.2.2.19](#Appendix_A_Target_82): In Windows, the default program is explorer.exe.

[<83> Section 2.2.2.19](#Appendix_A_Target_83): IPv6 is not supported in Windows NT, Windows 2000, Windows XP, or Windows Server 2003.

[<84> Section 2.2.2.19](#Appendix_A_Target_84): ColorDepth is not supported on Windows NT and Windows 2000.

The supported values 1, 2, 4, 8, 16, and 32 are translated, respectively, as the following number of colors supported: 16 (4 bpp), 256 (8 bpp), 65,536 (16 bpp), 16 million (24 bpp), 32,768 (15 bpp), and 16 million with transparency (32 bpp). A ColorDepth value of 32 indicates 32-bit color; this is not supported on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<85> Section 2.2.2.19](#Appendix_A_Target_85): ProtocolType is not supported on Windows NT or Windows 2000.

[<86> Section 2.2.2.19](#Appendix_A_Target_86): Not used by Terminal Services.

[<87> Section 2.2.2.19](#Appendix_A_Target_87): Not used by Terminal Services.

[<88> Section 2.2.2.19](#Appendix_A_Target_88): ClientTimeZone is not supported on Windows NT or Windows 2000.

[<89> Section 2.2.2.19](#Appendix_A_Target_89): ClientSessionId is not supported on Windows NT or Windows 2000.

[<90> Section 2.2.2.19](#Appendix_A_Target_90): clientDigProductId is not supported on Windows NT or Windows 2000.

If the terminal server client is either [**RDP**](#gt_17c795a6-68bf-46bf-8ea8-467c8df1a0b3) 5.0, 5.1, or 6.0 and is running on Windows XP or Windows Server 2003, this field contains the client machine's product ID from registry HKLM\Software\microsoft\windows\currentversion\productID. If RDP 6.0 terminal server client is running on a Windows Vista-based machine, it contains a unique instance ID of LSM process which is InstanceID in the TS registry.

[<91> Section 2.2.2.19](#Appendix_A_Target_91): PerformanceFlags is not supported on Windows NT or Windows 2000.

[<92> Section 2.2.2.19](#Appendix_A_Target_92): TS\_PERF\_ENABLE\_FONT\_SMOOTHING is not supported on Windows NT, Windows 2000, Windows XP, or Windows Server 2003.

[<93> Section 2.2.2.19](#Appendix_A_Target_93): TS\_PERF\_ENABLE\_DESKTOP\_COMPOSITION is not supported on Windows NT, Windows 2000, Windows XP, or Windows Server 2003.

[<94> Section 2.2.2.19](#Appendix_A_Target_94): ActiveInputLocale is not supported on Windows NT, Windows 2000, or Windows XP.

[<95> Section 2.2.2.20.1.3.1](#Appendix_A_Target_95): Not used by Terminal Services.

[<96> Section 2.2.2.21](#Appendix_A_Target_96): Not used by Terminal Services.

[<97> Section 2.2.2.21](#Appendix_A_Target_97): Not used by Terminal Services.

[<98> Section 2.2.2.27](#Appendix_A_Target_98): For the Microsoft RDP implementation, this driver is rdpwd.sys.

[<99> Section 2.2.2.27](#Appendix_A_Target_99): For the Microsoft RDP implementation, this binary is rdpwsx.dll.

[<100> Section 2.2.2.27](#Appendix_A_Target_100): WDF\_DYNAMIC\_RECONNECT is not supported on Windows NT or Windows 2000.

[<101> Section 2.2.2.27](#Appendix_A_Target_101): WDF\_SUBDESKTOP not supported on Windows NT or Windows 2000.

[<102> Section 2.2.2.27](#Appendix_A_Target_102): For the Microsoft RDP implementation, this binary is rdpcfgex.dll.

[<103> Section 2.2.2.27](#Appendix_A_Target_103): For the Microsoft RDP implementation, this string is "RDP".

[<104> Section 2.2.2.28](#Appendix_A_Target_104): Not supported on Windows NT or Windows 2000.

[<105> Section 2.2.2.31](#Appendix_A_Target_105): The POLICY\_TS\_MACHINE structure is supported only on Windows XP and Windows Server 2003.

[<106> Section 2.2.2.31](#Appendix_A_Target_106): fPolicyFipsEnabled is supported only on Windows XP and Windows Server 2003.

[<107> Section 2.2.2.31](#Appendix_A_Target_107): fPolicySecureLicensing is supported only on Windows XP and Windows Server 2003.

[<108> Section 2.2.2.31](#Appendix_A_Target_108): fPolicyEnableTimeZoneRedirection is not supported on Windows NT, Windows 2000, or Windows XP.

[<109> Section 2.2.2.31](#Appendix_A_Target_109): fPolicyDisableForcibleLogoff is not supported on Windows NT, Windows 2000, or Windows XP.

[<110> Section 2.2.2.31](#Appendix_A_Target_110): fPolicyLicensingMode is not supported on Windows NT, Windows 2000, or Windows XP.

[<111> Section 2.2.2.31](#Appendix_A_Target_111): fPolicyExplicitLSDiscovery is not supported on Windows NT, Windows 2000, or Windows XP.

[<112> Section 2.2.2.31](#Appendix_A_Target_112): fPolicyDisableTerminalServerTooltip is not supported on Windows NT, Windows 2000, or Windows XP.

[<113> Section 2.2.2.31](#Appendix_A_Target_113): ColorDepth is not supported on Windows NT or Windows 2000.

[<114> Section 2.2.2.31](#Appendix_A_Target_114): fSecureLicensing is not supported on Windows NT, Windows 2000, or Windows XP.

[<115> Section 2.2.2.31](#Appendix_A_Target_115): fEnableTimeZoneRedirection is not supported on Windows NT, Windows 2000, or Windows XP.

[<116> Section 2.2.2.31](#Appendix_A_Target_116): fDisableForcibleLogoff is supported only on Windows XP and Windows Server 2003.

[<117> Section 2.2.2.31](#Appendix_A_Target_117): fPolicyEncryptRPCTraffic is not supported on Windows NT, Windows 2000, or Windows XP.

[<118> Section 2.2.2.31](#Appendix_A_Target_118): fErrorInvalidProfile is not supported on Windows NT, Windows 2000, or Windows XP.

[<119> Section 2.2.2.31](#Appendix_A_Target_119): fPolicyFallbackPrintDriver is not supported on Windows NT, Windows 2000, or Windows XP.

[<120> Section 2.2.2.31](#Appendix_A_Target_120): FallbackPrintDriverType is not supported on Windows NT, Windows 2000, or Windows XP.

[<121> Section 2.2.2.31](#Appendix_A_Target_121): fDisableTerminalServerTooltip is not supported on Windows NT, Windows 2000, or Windows XP.

[<122> Section 2.2.2.31](#Appendix_A_Target_122): bSecurityLayer is not supported on Windows NT, Windows 2000, or Windows XP.

[<123> Section 2.2.2.31](#Appendix_A_Target_123): fPolicySecurityLayer is not supported on Windows NT, Windows 2000, or Windows XP.

[<124> Section 2.2.2.31](#Appendix_A_Target_124): fPolicyUserAuthentication is not supported on Windows NT, Windows 2000, or Windows XP.

[<125> Section 2.2.2.31](#Appendix_A_Target_125): fPolicyTurnOffSingleAppMode is not supported on Windows NT, Windows 2000, or Windows XP.

[<126> Section 2.2.2.31](#Appendix_A_Target_126): In Windows the default program is explorer.exe.

[<127> Section 2.2.2.33](#Appendix_A_Target_127): The value 32 in fColorDepth indicates 32-bit color; this is not supported on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<128> Section 2.2.2.35](#Appendix_A_Target_128): PROTOCOL\_CONSOLE is not supported in Windows NT or Windows 2000

[<129> Section 2.2.2.36](#Appendix_A_Target_129): DigProductId is not supported on Windows NT or Windows 2000. On Windows XP and Windows Server 2003, this field contains the machine's product ID from the registry key HKLM\Software\microsoft\windows\currentversion\productID.

[<130> Section 2.2.2.36](#Appendix_A_Target_130): ClientDigProductId is not supported on Windows NT or Windows 2000. If the terminal server client is RDP 5.0, 5.1, or 6.0 and is running on Windows XP or Windows Server 2003, this field contains the client machine's product ID from the registry key HKLM\Software\microsoft\windows\currentversion\productID. If an RDP 6.0 terminal server client is running on a Windows Vista-based machine, this field contains a unique instance ID of an LSM process, which is InstanceID in the TS registry.

[<131> Section 2.2.2.37](#Appendix_A_Target_131): IPv6 format addresses are not supported in Windows NT, Windows 2000, Windows XP, or Windows Server 2003.

[<132> Section 2.2.2.38](#Appendix_A_Target_132): Not used by Terminal Services.

[<133> Section 2.2.2.39](#Appendix_A_Target_133): For Terminal Services, this is icaapi.dll.

[<134> Section 2.2.2.39](#Appendix_A_Target_134): For Terminal Services, this is icaapi.dll.

[<135> Section 2.2.2.39](#Appendix_A_Target_135): For Terminal Services, this is termdd.sys.

[<136> Section 2.2.2.39](#Appendix_A_Target_136): For Terminal Services, this binary is rdpwd.sys.

[<137> Section 2.2.2.39](#Appendix_A_Target_137): Not used by Terminal Services.

[<138> Section 2.2.2.39](#Appendix_A_Target_138): For Terminal Services, this binary is rdpwd.sys.

[<139> Section 2.2.2.39](#Appendix_A_Target_139): For Terminal Services, this binary is tdtcp.sys.

[<140> Section 2.2.2.39](#Appendix_A_Target_140): Not used by Terminal Services.

[<141> Section 2.2.2.39](#Appendix_A_Target_141): Not used by Terminal Services.

[<142> Section 2.2.2.39](#Appendix_A_Target_142): Not used by Terminal Services.

[<143> Section 2.2.2.39](#Appendix_A_Target_143): For Terminal Services, this binary is rdpcfgex.dll.

[<144> Section 2.2.2.39](#Appendix_A_Target_144): For Terminal Services, this binary is rdpwsx.dll.

[<145> Section 2.2.2.39](#Appendix_A_Target_145): Not used by Terminal Services.

[<146> Section 2.2.2.44](#Appendix_A_Target_146): Not used by Terminal Services.

[<147> Section 2.3.1](#Appendix_A_Target_147): This field is not used by Terminal Services.

[<148> Section 3.1.1](#Appendix_A_Target_148): For Terminal Services, the **SECURITY\_DESCRIPTOR** for the console session is taken from the registry value HKLM\CurrentControlSet\Control\Terminal Server\WinStations\ConsoleSecurity. For remote sessions, it is taken from the registry value HKLM\CurrentControlSet\Control\Terminal Server\WinStations\ListenerName\Security, where ListenerName is the name of the listener through which the session was started when the connection request was received. If HKLM\CurrentControlSet\Control\Terminal Server\WinStations\ListenerName\Security does not exist or is empty, then it is taken from the registry value HKLM\CurrentControlSet\Control\Terminal Server\WinStations\DefaultSecurity.

[<149> Section 3.3.4.1.15](#Appendix_A_Target_149): Not supported on Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, or Windows Server 2008.

[<150> Section 3.3.4.1.16](#Appendix_A_Target_150): Not supported on Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, or Windows Server 2008.

[<151> Section 3.3.4.3.6](#Appendix_A_Target_151): *Level* is not supported on Windows NT, Windows 2000, Windows XP, or Windows Server 2003. The maximum value of *Level* supported is 2.

[<152> Section 3.3.4.3.9](#Appendix_A_Target_152): *Level* is not supported on Windows NT, Windows 2000, Windows XP, Windows Server 2003, or Windows Vista. The maximum supported value of *Level* is 2.

[<153> Section 3.3.4.3.10](#Appendix_A_Target_153): Not supported on Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, or Windows Server 2008.

[<154> Section 3.3.4.3.10](#Appendix_A_Target_154): Not supported on Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, or Windows Server 2008.

[<155> Section 3.3.4.3.11](#Appendix_A_Target_155): Not supported on Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows Server 2008, Windows Server 2008 R2, or Windows 7.

[<156> Section 3.5.4.1.2](#Appendix_A_Target_156): For information about setting permissions in Windows, see [[MSDN-Win32\_TSAcct]](https://go.microsoft.com/fwlink/?LinkId=119682).

[<157> Section 3.5.4.2.2](#Appendix_A_Target_157): The method returns an error code, E\_NOTIMPL (80004001), which is not supported on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<158> Section 3.7.4.1](#Appendix_A_Target_158): **[Opnums](#gt_e127848e-c66d-427d-b3aa-9f904fa4ada7)** reserved for local use apply to Windows as follows.

| Opnum | Description |
| --- | --- |
| 11 - 12 | Only used locally by Windows, never remotely. |
| 18 - 28 | Only used locally by Windows, never remotely. |
| 30 - 33 | Only used locally by Windows, never remotely. |
| 35 | Only used locally by Windows, never remotely. |
| 38 - 39 | Only used locally by Windows, never remotely. |
| 47 - 51 | Only used locally by Windows, never remotely. |
| 54 - 57 | Only used locally by Windows, never remotely. |
| 63 - 64 | Only used locally by Windows, never remotely. |
| 67 - 69 | Only used locally by Windows, never remotely. |
| 71 - 74 | Only used locally by Windows, never remotely. |

Opnums not implemented apply to Windows as follows.

| Opnum | Description |
| --- | --- |
| 29 | Exception is raised with error ERROR\_INVALID\_FUNCTION |
| 40 - 42 | Exception is raised with error ERROR\_INVALID\_FUNCTION |
| 52 | Exception is raised with error RPC\_S\_CANNOT\_SUPPORT |

All of these opnums return error code E\_NOTIMPL (0x80004001); this is not supported on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<159> Section 3.7.4.1.3](#Appendix_A_Target_159): [RpcIcaServerPing](#Section_fb4fa20ad77548b6b9b6edab02296225) is supported only on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<160> Section 3.7.4.1.5](#Appendix_A_Target_160): [RpcWinStationRename](#Section_84c75092c96a4beeac508f1632bc2f7b) is supported on Windows NT, Windows 2000, Windows XP, and Windows Server 2003. For information about setting permissions on Windows, see [MSDN-Win32\_TSAcct].

[<161> Section 3.7.4.1.6](#Appendix_A_Target_161): For information about setting permissions in Windows, see [MSDN-Win32\_TSAcct].

[<162> Section 3.7.4.1.6](#Appendix_A_Target_162): This information class is not supported on Windows NT and Windows 2000.

[<163> Section 3.7.4.1.6](#Appendix_A_Target_163): The driver is tdtcp.sys for Terminal Services.

[<164> Section 3.7.4.1.6](#Appendix_A_Target_164): The driver is rdpwd.sys for Terminal Services.

[<165> Section 3.7.4.1.6](#Appendix_A_Target_165): The driver is tdtcp.sys for Terminal Services.

[<166> Section 3.7.4.1.6](#Appendix_A_Target_166): This information class is not supported on Windows NT and Windows 2000.

[<167> Section 3.7.4.1.6](#Appendix_A_Target_167): This information class is not supported on Windows NT and Windows 2000.

[<168> Section 3.7.4.1.6](#Appendix_A_Target_168): This information class is not supported on Windows NT and Windows 2000.

[<169> Section 3.7.4.1.6](#Appendix_A_Target_169): This information class is not supported on Windows NT or Windows 2000.

[<170> Section 3.7.4.1.6](#Appendix_A_Target_170): This information class is supported only on Windows Server 2003.

[<171> Section 3.7.4.1.7](#Appendix_A_Target_171): This call is supported only on Windows NT 4.0 operating system, Windows 2000, Windows XP, and Windows Server 2003.

For information about setting permissions in Windows, see [MSDN-Win32\_TSAcct].

[<172> Section 3.7.4.1.7](#Appendix_A_Target_172): Ignored by the Microsoft terminal server drivers in Windows NT 4.0, Windows 2000, Windows XP, and Windows Server 2003.

[<173> Section 3.7.4.1.7](#Appendix_A_Target_173): Ignored by the Microsoft terminal server drivers in Windows NT 4.0, Windows 2000, Windows XP, and Windows Server 2003.

[<174> Section 3.7.4.1.7](#Appendix_A_Target_174): Used by Winlogon.

[<175> Section 3.7.4.1.8](#Appendix_A_Target_175): For information about setting permissions in Windows, see [MSDN-Win32\_TSAcct].

[<176> Section 3.7.4.1.9](#Appendix_A_Target_176): For information about setting permissions in Windows, see [MSDN-Win32\_TSAcct].

[<177> Section 3.7.4.1.10](#Appendix_A_Target_177): For information about setting permissions in Windows, see [MSDN-Win32\_TSAcct].

[<178> Section 3.7.4.1.11](#Appendix_A_Target_178): For information about setting permissions in Windows, see [MSDN-Win32\_TSAcct].

[<179> Section 3.7.4.1.11](#Appendix_A_Target_179): This parameter is supported only on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<180> Section 3.7.4.1.12](#Appendix_A_Target_180): For information about setting permissions in Windows, see [MSDN-Win32\_TSAcct].

[<181> Section 3.7.4.1.12](#Appendix_A_Target_181): This parameter is supported only on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<182> Section 3.7.4.1.13](#Appendix_A_Target_182): In the case of resetting a [**listener**](#gt_1e5aa0c2-4617-4d7a-969e-a7091c415c8d), all sessions started by Terminal Services based on connection requests received through that listener will be disconnected; this is not supported on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

For information about setting permissions in Windows, see [MSDN-Win32\_TSAcct].

[<183> Section 3.7.4.1.15](#Appendix_A_Target_183): Not used by Terminal Services.

[<184> Section 3.7.4.1.16](#Appendix_A_Target_184): This call is supported only on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<185> Section 3.7.4.1.17](#Appendix_A_Target_185): This call is supported only on Windows NT and Windows 2000.

In Windows the session manager is csrss.exe.

[<186> Section 3.7.4.1.17](#Appendix_A_Target_186): On Windows 2000, the method ignores any value for *LogonId* other than -2 and will not allow the debugger to be broken into in kernel mode or in a particular session's csrss.exe process.

[<187> Section 3.7.4.1.18](#Appendix_A_Target_187): The registry reread is HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations for each subkey under [**WinStation**](#gt_8cc89bb9-be94-4672-8a5d-9947b91bc952).

[<188> Section 3.7.4.1.18](#Appendix_A_Target_188): This method is supported only on Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<189> Section 3.7.4.1.19](#Appendix_A_Target_189): The [OldRpcWinStationEnumerateProcesses](#Section_f2b9af5b53494f4f9d8416e5271a04d9) method returns the process information for a terminal server and is supported only on Windows NT Server 4.0 operating system terminal server for backward compatibility.

[<190> Section 3.7.4.1.20](#Appendix_A_Target_190): The [RpcWinStationEnumerateProcesses](#Section_3559bc198ca34bcf842ecf25f592dc1a) method returns the process information for a terminal server and is supported only on Windows NT Server 4.0 terminal server for backwards compatibility.

[<191> Section 3.7.4.1.23](#Appendix_A_Target_191): RpcWinStationGetProcessSid is not supported in Windows NT.

[<192> Section 3.7.4.1.23](#Appendix_A_Target_192): This value is not returned in Windows NT, Windows 2000, Windows XP, or Windows Server 2003.

[<193> Section 3.7.4.1.23](#Appendix_A_Target_193): This value is returned only in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<194> Section 3.7.4.1.23](#Appendix_A_Target_194): This value is not returned in Windows NT, Windows 2000, Windows XP, or Windows Server 2003.

[<195> Section 3.7.4.1.23](#Appendix_A_Target_195): This value is returned only in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<196> Section 3.7.4.1.23](#Appendix_A_Target_196): This value is returned only in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<197> Section 3.7.4.1.23](#Appendix_A_Target_197): This value is not returned in Windows NT, Windows 2000, Windows XP, or Windows Server 2003.

[<198> Section 3.7.4.1.23](#Appendix_A_Target_198): This value is returned only in Windows NT, Windows 2000, Windows XP, and Windows Server 2003.

[<199> Section 3.7.4.1.23](#Appendix_A_Target_199): This value is not returned in Windows NT, Windows 2000, Windows XP, or Windows Server 2003.

[<200> Section 3.7.4.1.25](#Appendix_A_Target_200): Done for all WinStations specified in the registry under HKLM\System\CurrentControlSet\Terminal Server\WinStations.

[<201> Section 3.7.4.1.25](#Appendix_A_Target_201): This call is supported on Windows 2000, Windows XP, and Windows Server 2003.

[<202> Section 3.7.4.1.26](#Appendix_A_Target_202): The GUIDs for the network adapters implementing terminal server are found as subkeys under the registry key HKLM\System\CurrentControlSet\Control\Terminal Server\lanatable, where each key is the adapter's GUID, and the DWORD value LanaId under that key is the lana that is matched to the parameter LanAdapter.

[<203> Section 3.7.4.1.26](#Appendix_A_Target_203): This call is supported only on Windows 2000, Windows XP, and Windows Server 2003.

[<204> Section 3.7.4.1.27](#Appendix_A_Target_204): RpcWinStationUpdateSettings is supported only in Windows XP and Windows Server 2003.

In Windows XP, a class of WINSTACFG\_SESSDIR will do nothing and always returns success.

[<205> Section 3.7.4.1.27](#Appendix_A_Target_205): If running in Remote Desktop mode (Windows XP) or in Remote Admin mode, this does nothing.

[<206> Section 3.7.4.1.28](#Appendix_A_Target_206): RpcWinStationShadowStop is supported only in Windows XP and Windows Server 2003.

[<207> Section 3.7.4.1.30](#Appendix_A_Target_207): RpcWinStationIsHelpAssistantSession is supported only in Windows XP and Windows Server 2003.

[<208> Section 3.7.4.1.31](#Appendix_A_Target_208): RpcWinStationGetMachinePolicy is supported only on Windows XP and Windows Server 2003.

[<209> Section 3.7.4.1.32](#Appendix_A_Target_209): RpcWinStationCheckLoopBack is supported only on Windows XP and Windows Server 2003.

[<210> Section 3.7.4.1.33](#Appendix_A_Target_210): RpcConnectCallback is supported only in Windows XP and Windows Server 2003.

[<211> Section 3.7.4.1.35](#Appendix_A_Target_211): Supported on Windows Server 2003 only, and only on advanced servers running in [**application server mode**](#gt_b95e0f34-13ff-496c-8cad-bf2fe26a5b59).

[<212> Section 3.9.4.1.1](#Appendix_A_Target_212): This call is not supported on Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, or Windows Server 2008.

# 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

client ([section 3.2.1](#section_8d8b6e48931a4ee6b57da2837bd86f80) 95, [section 3.4.1](#section_2c58ef6bd66544fc952df7bda2f73521) 119, [section 3.6.1](#section_36afd60871654f458a89a81f98ea5223) 131)

[legacy](#section_36afd60871654f458a89a81f98ea5223) 131

[local session manager](#section_8d8b6e48931a4ee6b57da2837bd86f80) 95

[TermSrv](#section_2c58ef6bd66544fc952df7bda2f73521) 119

[virtual IP](#section_c4f9189db9e54aa6b8c4f24850380ad1) 175

server ([section 3.3.1](#section_fc7b89539126492f95ea1554b2023fce) 95, [section 3.5.1](#section_b164fe5545f045da852bc79fffc1954b) 119, [section 3.7.1](#section_29c1c1433b824d4a880aa01f0274762c) 131)

[legacy](#section_29c1c1433b824d4a880aa01f0274762c) 131

[local session manager](#section_fc7b89539126492f95ea1554b2023fce) 95

[TermSrv](#section_b164fe5545f045da852bc79fffc1954b) 119

[virtual IP](#section_fe9fa64a3df14280ae3e1ac4812eae3a) 176

[Access rights - caller - overview](#section_2c11ff9c9319444ca220653506edb65c) 94

[Applicability](#section_53af874602bf45d580713a330564e330) 15

[appsrv\tsvip\rpc\tsviprpc.idl](#section_e8228308e1ba4a5586fbcb4bb5b3626d) 207

[ASYNCCONFIG structure](#section_f6c52c7c88614ce688f18c84e4f2026a) 59

[ASYNCCONNECTCLASS enumeration](#section_4e0d193c1b744ba098cc33255405b3a5) 62

B

[BEEPINPUT structure](#section_2faf4c5b3a79491a9d1a145f46a797d1) 80

C

[CACHE\_STATISTICS structure](#section_81203ca2e58b4681affa924e59671b5c) 57

[CALLBACKCLASS enumeration](#section_4fa6b4b0cc9c46c88c2068013a645e92) 48

[Caller permissions and access rights - overview](#section_2c11ff9c9319444ca220653506edb65c) 94

[Capability negotiation](#section_3a25197a73e04ffb97619ab8971cff26) 16

[CDCLASS enumeration](#section_ad6aac8900554cbf986825cb944fe294) 66

[CDCONFIG structure](#section_f88900e1c1594f02b3ae84f05eec212f) 66

[Change tracking](#section_c1079a6754f348e58b57765a499670ce) 243

Client

abstract data model ([section 3.2.1](#section_8d8b6e48931a4ee6b57da2837bd86f80) 95, [section 3.4.1](#section_2c58ef6bd66544fc952df7bda2f73521) 119, [section 3.6.1](#section_36afd60871654f458a89a81f98ea5223) 131)

initialization ([section 3.2.3](#section_0e0decb95cac47a5b57713c1793d6dc4) 95, [section 3.4.3](#section_880c60d039a24111a8f645df85df405d) 119, [section 3.6.3](#section_84a1056f0c364341bbd3295da540e3b7) 131)

legacy

[abstract data model](#section_36afd60871654f458a89a81f98ea5223) 131

[initialization](#section_84a1056f0c364341bbd3295da540e3b7) 131

[local events](#section_cd42194547804c2383dbeb0b51cf48f8) 131

[message processing](#section_42c14d8f56a64444a194e9586f145c56) 131

[sequencing rules](#section_42c14d8f56a64444a194e9586f145c56) 131

[timer events](#section_566e82d44ea74a37ad805774304ba69c) 131

[timers](#section_215729887a554f9890fb9088bdb68108) 131

local events ([section 3.2.6](#section_579414f868e843fb8986a6f01f6d3d87) 95, [section 3.4.6](#section_b8cea0b6d52747f39a8d3ca470a9385a) 119, [section 3.6.6](#section_cd42194547804c2383dbeb0b51cf48f8) 131)

local session manager

[abstract data model](#section_8d8b6e48931a4ee6b57da2837bd86f80) 95

[initialization](#section_0e0decb95cac47a5b57713c1793d6dc4) 95

[local events](#section_579414f868e843fb8986a6f01f6d3d87) 95

[message processing](#section_3f92d2e1f9574e7580f3ea700851535c) 95

[sequencing rules](#section_3f92d2e1f9574e7580f3ea700851535c) 95

[timer events](#section_7b909a4657b14fa7adfc119334458beb) 95

[timers](#section_f20440c6f3cd4e9786f443cfa8cb5044) 95

[message processing](#section_42c14d8f56a64444a194e9586f145c56) 131

[sequencing rules](#section_42c14d8f56a64444a194e9586f145c56) 131

TermSrv

[abstract data model](#section_2c58ef6bd66544fc952df7bda2f73521) 119

[initialization](#section_880c60d039a24111a8f645df85df405d) 119

[local events](#section_b8cea0b6d52747f39a8d3ca470a9385a) 119

[message processing](#section_04c44f66db064ac28c2b9a7bc2e9f2af) 119

[sequencing rules](#section_04c44f66db064ac28c2b9a7bc2e9f2af) 119

[timer events](#section_7b2bf211c7a74f79b6967f391e3eb1e4) 119

[timers](#section_7128dcf9ea4d46aab671822f35d4deab) 119

timer events ([section 3.2.5](#section_7b909a4657b14fa7adfc119334458beb) 95, [section 3.4.5](#section_7b2bf211c7a74f79b6967f391e3eb1e4) 119, [section 3.6.5](#section_566e82d44ea74a37ad805774304ba69c) 131)

timers ([section 3.2.2](#section_f20440c6f3cd4e9786f443cfa8cb5044) 95, [section 3.4.2](#section_7128dcf9ea4d46aab671822f35d4deab) 119, [section 3.6.2](#section_215729887a554f9890fb9088bdb68108) 131)

virtual IP

[abstract data model](#section_c4f9189db9e54aa6b8c4f24850380ad1) 175

[initialization](#section_d8e5648f3b9a428a91d3b6abd5603152) 175

[local events](#section_4865f6f4547740859fedba1174e043f6) 176

[message processing](#section_26a3d2a761224b7d99767e7200e4d447) 176

[sequencing rules](#section_26a3d2a761224b7d99767e7200e4d447) 176

[timer events](#section_39b146adc1544e5ebab07b4c97b712e8) 176

[timers](#section_49bf96df07a1467eb9496ed42658e987) 175

[CLIENT\_ID structure](#section_a11e7129685b45358d3721d4596ac057) 43

[CLIENT\_STACK\_ADDRESS structure](#section_78526580c032463389bb17a0a49e2b4a) 82

[Common data types](#section_f1a460a6c19e4ad2a1f6d67a71732262) 17

[CONNECTCONFIG structure](#section_8f14c17baf564430bcab9231d1334b29) 61

D

Data model - abstract

client ([section 3.2.1](#section_8d8b6e48931a4ee6b57da2837bd86f80) 95, [section 3.4.1](#section_2c58ef6bd66544fc952df7bda2f73521) 119, [section 3.6.1](#section_36afd60871654f458a89a81f98ea5223) 131)

[legacy](#section_36afd60871654f458a89a81f98ea5223) 131

[local session manager](#section_8d8b6e48931a4ee6b57da2837bd86f80) 95

[TermSrv](#section_2c58ef6bd66544fc952df7bda2f73521) 119

[virtual IP](#section_c4f9189db9e54aa6b8c4f24850380ad1) 175

server ([section 3.3.1](#section_fc7b89539126492f95ea1554b2023fce) 95, [section 3.5.1](#section_b164fe5545f045da852bc79fffc1954b) 119, [section 3.7.1](#section_29c1c1433b824d4a880aa01f0274762c) 131)

[legacy](#section_29c1c1433b824d4a880aa01f0274762c) 131

[local session manager](#section_fc7b89539126492f95ea1554b2023fce) 95

[TermSrv](#section_b164fe5545f045da852bc79fffc1954b) 119

[virtual IP](#section_fe9fa64a3df14280ae3e1ac4812eae3a) 176

Data types

[common - overview](#section_f1a460a6c19e4ad2a1f6d67a71732262) 17

[structures](#section_5dbf8683a41343988dd34e0122d2443b) 27

[Decoding/encoding example](#section_5b5aff03b8b74f9eb15fd4cbe3f03d4e) 189

[Determining caller permissions and access rights - overview](#section_2c11ff9c9319444ca220653506edb65c) 94

[Directory service schema elements](#section_9b65ec6ba9c7401fbc4a0f20538004a1) 86

E

[Elements - directory service schema](#section_9b65ec6ba9c7401fbc4a0f20538004a1) 86

[Encoding/decoding example](#section_5b5aff03b8b74f9eb15fd4cbe3f03d4e) 189

[Encoding/decoding example example](#section_5b5aff03b8b74f9eb15fd4cbe3f03d4e) 189

Events

local

[legacy client](#section_cd42194547804c2383dbeb0b51cf48f8) 131

[legacy server](#section_b30c369ccba64f20a6ae4df50a021211) 175

[LSM client](#section_579414f868e843fb8986a6f01f6d3d87) 95

[LSM server](#section_06a8eb514215444da30185cd71adf259) 119

[TermSrv client](#section_b8cea0b6d52747f39a8d3ca470a9385a) 119

[TermSrv server](#section_94365f4f376c4bdb876b8cfe80bd0921) 131

[virtual IP client](#section_4865f6f4547740859fedba1174e043f6) 176

[virtual IP server](#section_20297b8e7b8940b8abfefaee7fba8b07) 177

local - client ([section 3.2.6](#section_579414f868e843fb8986a6f01f6d3d87) 95, [section 3.4.6](#section_b8cea0b6d52747f39a8d3ca470a9385a) 119, [section 3.6.6](#section_cd42194547804c2383dbeb0b51cf48f8) 131)

local - server ([section 3.3.6](#section_06a8eb514215444da30185cd71adf259) 119, [section 3.5.6](#section_94365f4f376c4bdb876b8cfe80bd0921) 131, [section 3.7.6](#section_b30c369ccba64f20a6ae4df50a021211) 175)

timer

[legacy client](#section_566e82d44ea74a37ad805774304ba69c) 131

[legacy server](#section_34d19b4fb0e84dddab178ede7c20a310) 175

[LSM client](#section_7b909a4657b14fa7adfc119334458beb) 95

[LSM server](#section_dede264ad2dc460d9681f32d4a013885) 119

[TermSrv client](#section_7b2bf211c7a74f79b6967f391e3eb1e4) 119

[TermSrv server](#section_79450165a34d4bdb93986e33b20c7970) 130

[virtual IP client](#section_39b146adc1544e5ebab07b4c97b712e8) 176

[virtual IP server](#section_a290eb1fec724285944091a5af84e42a) 177

timer - client ([section 3.2.5](#section_7b909a4657b14fa7adfc119334458beb) 95, [section 3.4.5](#section_7b2bf211c7a74f79b6967f391e3eb1e4) 119, [section 3.6.5](#section_566e82d44ea74a37ad805774304ba69c) 131)

timer - server ([section 3.3.5](#section_dede264ad2dc460d9681f32d4a013885) 119, [section 3.5.5](#section_79450165a34d4bdb93986e33b20c7970) 130, [section 3.7.5](#section_34d19b4fb0e84dddab178ede7c20a310) 175)

Examples

[ecoding/decoding](#section_5b5aff03b8b74f9eb15fd4cbe3f03d4e) 189

[encoding/decoding example](#section_5b5aff03b8b74f9eb15fd4cbe3f03d4e) 189

[legacy](#section_2c7cc64df36747919d7497d0c3d38671) 187

[legacy example](#section_2c7cc64df36747919d7497d0c3d38671) 187

[LSM enumeration](#section_d67ea0b012604edb9f3c417d50011d23) 180

[lsm enumeration example](#section_d67ea0b012604edb9f3c417d50011d23) 180

[TermService listener](#section_d045ce70be964d2392a45b842893b332) 182

[termservice listener example](#section_d045ce70be964d2392a45b842893b332) 182

[TermSrvBindSecure](#section_bc8c6536baff466bb12e435b66577bbd) 184

[termsrvbindsecure example](#section_bc8c6536baff466bb12e435b66577bbd) 184

[EXECENVDATA structure](#section_20bf286ea1584d5fa9f495eeaf1a4869) 31

[EXECENVDATA\_LEVEL1 structure](#section_004305a642654ff5bb8a58fd0f7cf2d5) 32

[EXECENVDATA\_LEVEL2 structure](#section_e85daed7ed774c8aafc05596c8d41c9c) 32

[EXECENVDATAEX structure](#section_0a3f7e7b03f84f62a65fc2d670bb3173) 33

[EXECENVDATAEX\_LEVEL1 structure](#section_b4295b0ec64140abbbdbea44323b0804) 33

[ExtendedClientCredentials structure](#section_280cd64d389d4c8fb5fccf429abc5d88) 77

F

[Fields - vendor-extensible](#section_0d8eb7e164594729bc72320557ddff32) 16

[FLOWCONTROLCLASS enumeration](#section_e5ba41c8f4e04c69a411715571d5f389) 60

[FLOWCONTROLCONFIG structure](#section_aba2eab5c9b848aabefcae8b32a3efd7) 60

[Full IDL](#section_c43addc7eebc491b9b012587262675e8) 192

G

[Glossary](#section_c41d336704c94c93babf9b5de834eb29) 10

I

[IDL](#section_c43addc7eebc491b9b012587262675e8) 192

[Implementation - overview](#section_fb6aeeac34d7426eacdae198581443fe) 94

[Implementer - security considerations](#section_81b1b9b4338040b9ab06bb1b20cbf2e4) 191

[Index of security parameters](#section_3ecc8b1f5f7f46f48c47ddd7594a4f70) 191

[Informative references](#section_2ccce14446044fa8ac10695d91e459f4) 14

Initialization

client ([section 3.2.3](#section_0e0decb95cac47a5b57713c1793d6dc4) 95, [section 3.4.3](#section_880c60d039a24111a8f645df85df405d) 119, [section 3.6.3](#section_84a1056f0c364341bbd3295da540e3b7) 131)

[legacy](#section_84a1056f0c364341bbd3295da540e3b7) 131

[local session manager](#section_0e0decb95cac47a5b57713c1793d6dc4) 95

[TermSrv](#section_880c60d039a24111a8f645df85df405d) 119

[virtual IP](#section_d8e5648f3b9a428a91d3b6abd5603152) 175

server ([section 3.3.3](#section_de25f861bfb94f68aea51026292bbfb3) 98, [section 3.5.3](#section_f1ba74cb7e5340fba255e43fbc00def0) 121, [section 3.7.3](#section_caa1aed79f37455781da8e44fb5d3382) 134)

[legacy](#section_caa1aed79f37455781da8e44fb5d3382) 134

[local session manager](#section_de25f861bfb94f68aea51026292bbfb3) 98

[TermSrv](#section_f1ba74cb7e5340fba255e43fbc00def0) 121

[virtual IP](#section_7443fec9e2094eb8a4152079cd0deab9) 176

[Introduction](#section_a4c83616dc4c4c42bc411ace892e4684) 10

L

Legacy client

[abstract data model](#section_36afd60871654f458a89a81f98ea5223) 131

[initialization](#section_84a1056f0c364341bbd3295da540e3b7) 131

[local events](#section_cd42194547804c2383dbeb0b51cf48f8) 131

[message processing](#section_42c14d8f56a64444a194e9586f145c56) 131

[sequencing rules](#section_42c14d8f56a64444a194e9586f145c56) 131

[timer events](#section_566e82d44ea74a37ad805774304ba69c) 131

[timers](#section_215729887a554f9890fb9088bdb68108) 131

[Legacy example](#section_2c7cc64df36747919d7497d0c3d38671) 187

[Legacy example example](#section_2c7cc64df36747919d7497d0c3d38671) 187

Legacy server

[abstract data model](#section_29c1c1433b824d4a880aa01f0274762c) 131

[initialization](#section_caa1aed79f37455781da8e44fb5d3382) 134

[local events](#section_b30c369ccba64f20a6ae4df50a021211) 175

[message processing](#section_2268c9823653425c86628f038a8f4f2f) 134

[sequencing rules](#section_2268c9823653425c86628f038a8f4f2f) 134

[timer events](#section_34d19b4fb0e84dddab178ede7c20a310) 175

[timers](#section_550cbf95ef0f446b8bddff21af420aa2) 134

[LegacyApi method](#section_d9f6ec940483417695325b47455133ed) 134

[LISTENERENUM structure](#section_3c46804ffff447799f961ceb9a6c9559) 36

[LISTENERENUM\_LEVEL1 structure](#section_810c8c1548e342229c5058cd21c729fb) 37

[LOADFACTORTYPE enumeration](#section_2d4b1921a38d4dea996ca4470a0fe9db) 75

Local events

client ([section 3.2.6](#section_579414f868e843fb8986a6f01f6d3d87) 95, [section 3.4.6](#section_b8cea0b6d52747f39a8d3ca470a9385a) 119, [section 3.6.6](#section_cd42194547804c2383dbeb0b51cf48f8) 131)

[legacy](#section_cd42194547804c2383dbeb0b51cf48f8) 131

[local session manager](#section_579414f868e843fb8986a6f01f6d3d87) 95

[TermSrv](#section_b8cea0b6d52747f39a8d3ca470a9385a) 119

[virtual IP](#section_4865f6f4547740859fedba1174e043f6) 176

server ([section 3.3.6](#section_06a8eb514215444da30185cd71adf259) 119, [section 3.5.6](#section_94365f4f376c4bdb876b8cfe80bd0921) 131, [section 3.7.6](#section_b30c369ccba64f20a6ae4df50a021211) 175)

[legacy](#section_b30c369ccba64f20a6ae4df50a021211) 175

[local session manager](#section_06a8eb514215444da30185cd71adf259) 119

[TermSrv](#section_94365f4f376c4bdb876b8cfe80bd0921) 131

[virtual IP](#section_20297b8e7b8940b8abfefaee7fba8b07) 177

LSM client

[abstract data model](#section_8d8b6e48931a4ee6b57da2837bd86f80) 95

[initialization](#section_0e0decb95cac47a5b57713c1793d6dc4) 95

[local events](#section_579414f868e843fb8986a6f01f6d3d87) 95

[message processing](#section_3f92d2e1f9574e7580f3ea700851535c) 95

[sequencing rules](#section_3f92d2e1f9574e7580f3ea700851535c) 95

[timer events](#section_7b909a4657b14fa7adfc119334458beb) 95

[timers](#section_f20440c6f3cd4e9786f443cfa8cb5044) 95

[LSM enumeration example](#section_d67ea0b012604edb9f3c417d50011d23) 180

[Lsm enumeration example example](#section_d67ea0b012604edb9f3c417d50011d23) 180

LSM server

[abstract data model](#section_fc7b89539126492f95ea1554b2023fce) 95

[initialization](#section_de25f861bfb94f68aea51026292bbfb3) 98

[local events](#section_06a8eb514215444da30185cd71adf259) 119

[message processing](#section_103be5dd283b4a41bf030430484ff583) 98

[sequencing rules](#section_103be5dd283b4a41bf030430484ff583) 98

[timer events](#section_dede264ad2dc460d9681f32d4a013885) 119

[timers](#section_47fb618362474c94adb598eb067771d7) 98

[LSM\_SESSIONINFO\_EX\_LEVEL1 structure](#section_2789ef970c7c4ec59f2cd375de4941a0) 35

[LSMSESSIONINFORMATION structure](#section_fadef65b0d6542b8ad14127106499f85) 34

[LSMSESSIONINFORMATION\_EX structure](#section_4e7039f4d1d044f282f008b057476db5) 35

M

Message processing

[client](#section_42c14d8f56a64444a194e9586f145c56) 131

[legacy](#section_42c14d8f56a64444a194e9586f145c56) 131

[local session manager](#section_3f92d2e1f9574e7580f3ea700851535c) 95

[TermSrv](#section_04c44f66db064ac28c2b9a7bc2e9f2af) 119

[virtual IP](#section_26a3d2a761224b7d99767e7200e4d447) 176

server ([section 3.5.4](#section_24e967207f93438e996e724b2d56ce4c) 121, [section 3.7.4](#section_2268c9823653425c86628f038a8f4f2f) 134)

[legacy](#section_2268c9823653425c86628f038a8f4f2f) 134

[local session manager](#section_103be5dd283b4a41bf030430484ff583) 98

[TermSrv](#section_24e967207f93438e996e724b2d56ce4c) 121

[virtual IP](#section_0089cf899f8345f081036c261c8ef302) 176

Messages

[common data types](#section_f1a460a6c19e4ad2a1f6d67a71732262) 17

[transport](#section_ac0535d0a8474386ae3ee93a50ff2484) 17

Methods

[LegacyApi](#section_d9f6ec940483417695325b47455133ed) 134

[RCMListener](#section_40742e8650ac4bb696a619e629a06aff) 128

[RCMPublic](#section_3cfda13d57b64161a75aa654c9bc7428) 121

[msTSProperty01 attribute](#section_8c87e949691c4083b945fdf12e21585f) 93

N

[NASICONFIG structure](#section_bd57a02de0834a22946a581d939dd6dc) 62

[NETWORKCONFIG structure](#section_a962dcf904ae461aacf14b50e677c265) 59

[Normative references](#section_202a2f5f8297429483efb7b8e92fa3f1) 12

[NT6\_TS\_UNICODE\_STRING structure](#section_29ce6198bc1d42fe8b2c87287d87edcf) 42

O

[OEMTDCONFIG structure](#section_352301477fe047d893d41ace099fa8e0) 63

[OldRpcWinStationEnumerateProcesses method](#section_f2b9af5b53494f4f9d8416e5271a04d9) 160

[Overview (synopsis)](#section_0fb8dffb3d8243ca8c096f62e36cfc9a) 14

P

[Parameter index - security](#section_3ecc8b1f5f7f46f48c47ddd7594a4f70) 191

[Parameters - security index](#section_3ecc8b1f5f7f46f48c47ddd7594a4f70) 191

[PASYNCCONFIG](#section_f6c52c7c88614ce688f18c84e4f2026a) 59

[PBEEPINPUT](#section_2faf4c5b3a79491a9d1a145f46a797d1) 80

[PCACHE\_STATISTICS](#section_81203ca2e58b4681affa924e59671b5c) 57

[PCDCONFIG](#section_f88900e1c1594f02b3ae84f05eec212f) 66

[PCLIENT\_STACK\_ADDRESS](#section_78526580c032463389bb17a0a49e2b4a) 82

[PCONNECTCONFIG](#section_8f14c17baf564430bcab9231d1334b29) 61

[PDCONFIG structure](#section_6fe0f4f264874390890ee7b0e4814d29) 63

[PDCONFIG2 structure](#section_74204022eb7c4454b3d024f642c892a4) 63

[PDPARAMS structure](#section_ff0e2998fb8b4dffab648427fad556eb) 58

[PDPARAMSWIRE structure](#section_2e867f3dfc4d4ff6a01fec5d3c1173fa) 82

[Permissions - caller - overview](#section_2c11ff9c9319444ca220653506edb65c) 94

[PEXECENVDATA](#section_20bf286ea1584d5fa9f495eeaf1a4869) 31

[PEXECENVDATA\_LEVEL1](#section_004305a642654ff5bb8a58fd0f7cf2d5) 32

[PEXECENVDATA\_LEVEL2](#section_e85daed7ed774c8aafc05596c8d41c9c) 32

[PEXECENVDATAEX](#section_0a3f7e7b03f84f62a65fc2d670bb3173) 33

[PEXECENVDATAEX\_LEVEL1](#section_b4295b0ec64140abbbdbea44323b0804) 33

[pExtendedClientCredentials](#section_280cd64d389d4c8fb5fccf429abc5d88) 77

[PFLOWCONTROLCONFIG](#section_aba2eab5c9b848aabefcae8b32a3efd7) 60

[PLISTENERENUM](#section_3c46804ffff447799f961ceb9a6c9559) 36

[PLISTENERENUM\_LEVEL1](#section_810c8c1548e342229c5058cd21c729fb) 37

[PLSM\_SESSIONINFO\_EX\_LEVEL1](#section_2789ef970c7c4ec59f2cd375de4941a0) 35

[PLSMSESSIONINFORMATION](#section_fadef65b0d6542b8ad14127106499f85) 34

[PLSMSESSIONINFORMATION\_EX](#section_4e7039f4d1d044f282f008b057476db5) 35

[PNASICONFIG](#section_bd57a02de0834a22946a581d939dd6dc) 62

[PNETWORKCONFIG](#section_a962dcf904ae461aacf14b50e677c265) 59

[POEMTDCONFIG](#section_352301477fe047d893d41ace099fa8e0) 63

[POLICY\_TS\_MACHINE structure](#section_d7cb8c2df60e42d6b8f4e617ad4d8c1b) 68

[PPDCONFIG](#section_6fe0f4f264874390890ee7b0e4814d29) 63

[PPDCONFIG2](#section_74204022eb7c4454b3d024f642c892a4) 63

[PPDPARAMS](#section_ff0e2998fb8b4dffab648427fad556eb) 58

[PPDPARAMSWIRE](#section_2e867f3dfc4d4ff6a01fec5d3c1173fa) 82

[PPOLICY\_TS\_MACHINE](#section_d7cb8c2df60e42d6b8f4e617ad4d8c1b) 68

[PPROTOCOLCOUNTERS](#section_c5d7f335bfd9444d87a5b0d028a52b5e) 56

[PPROTOCOLSTATUS](#section_c9066753acbd46789a728fb1b080bd09) 55

[PPROTOCOLSTATUSEX](#section_e32b64ed22e245cab9549e1b52a16c78) 56

[PRCM\_REMOTEADDRESS](#section_3c5dea7cf9a441c6b9f831b5bc985b6b) 81

[Preconditions](#section_604142508d87489db053f3fc51743d48) 15

[Prerequisites](#section_604142508d87489db053f3fc51743d48) 15

[PRESERVED\_CACHE](#section_9c0c0474ac364315b0ba8fca36b5d6e5) 58

[Product behavior](#section_7ce7f717461b4f1898296690168c7707) 230

[PropValue field - encoding in TSProperty structure](#section_27cd734ab78041e6b9d5b6c9fba6c16a) 92

Protocol Details

[overview](#section_fb6aeeac34d7426eacdae198581443fe) 94

[PROTOCOLCOUNTERS structure](#section_c5d7f335bfd9444d87a5b0d028a52b5e) 56

[PROTOCOLSTATUS structure](#section_c9066753acbd46789a728fb1b080bd09) 55

[PROTOCOLSTATUS\_INFO\_TYPE enumeration](#section_dd616139394c4ac281c7175024327604) 27

[PROTOCOLSTATUSEX structure](#section_e32b64ed22e245cab9549e1b52a16c78) 56

[PSESSION\_CHANGE](#section_e22785c95a374a47864aed2865d053bc) 81

[PSESSIONENUM](#section_fd8759ef4f6c45c6b5429d5e53e90f7d) 28

[PSESSIONENUM\_EX](#section_35a722eb00a04ba08e4c735d06378675) 30

[PSESSIONENUM\_LEVEL1](#section_0c4fde3b4fdb45248cae792d0320247a) 29

[PSESSIONENUM\_LEVEL2](#section_1b889bfe8d324cbabd92f9d7c09d5f60) 29

[PSESSIONENUM\_LEVEL3](#section_d1662f979e3a4fbbbdd52050df69e902) 29

[PSESSIONID](#section_e5c755e86d66494d873f4540267fe2f4) 37

[PSYSTEM\_THREAD\_INFORMATION](#section_e82d73e4cedb40779099d58f3459722f) 42

[PTHINWIRECACHE](#section_3adaf1a0f6324440aeca103d9e4097ea) 58

[PTS\_ALL\_PROCESSES\_INFO](#section_8f5564a61b5e4381afc44acf7ad39d6d) 38

[PTS\_ALL\_PROCESSES\_INFO\_NT6](#section_d60fc5882cc744649c53879142f5a58d) 40

[PTS\_COUNTER](#section_a0173488c4834340ab1aaab2d21ea323) 43

[PTS\_COUNTER\_HEADER](#section_e08317cba69340d39a5f61e14cfdb431) 44

[PTS\_PROCESS\_INFORMATION\_NT4](#section_0a8a73d93c9e4d2d91e47fef48b6c29d) 38

[PTS\_SYS\_PROCESS\_INFORMATION](#section_96702c7d6f5f49658f295fdbea73278e) 38

[PTS\_SYS\_PROCESS\_INFORMATION\_NT6](#section_66ef46a7504f469696130bd8446ee225) 41

[PTS\_TRACE](#section_ad34957553694830a1741920a7af8d5f) 78

[PTSHARE\_CACHE](#section_07167f99bd894338b3c77aafcc75334f) 58

[PTSHARE\_COUNTERS](#section_9981de84885c4ff8a8709912a8a5ff12) 57

[PTSVIP\_SOCKADDR](#section_53383eb1f61f401389905a5c7b95e10b) 83

[PTSVIPAddress](#section_eb4b65abdbfe481db85f7931ee38c2cc) 84

[PTSVIPSession](#section_92043576fad1495c9a5df90185959e5f) 85

[PUSERCONFIG](#section_dba750b8cb354e889811e2a1f8a10701) 45

[PVARDATA\_WIRE](#section_cc15eeb5f98148d193f6d4f9bfdb07ca) 82

[PWDCONFIG](#section_d1bf099beb544ed9a7230b1062dbc128) 65

[PWINSTACONFIGWIRE](#section_ae5355e7c6cc48abb82ad1de8b3cd6d6) 83

[PWINSTATIONCLIENT](#section_673d8ac0f55748cb98a649925160d729) 49

[PWINSTATIONCLIENTDATA](#section_d96b68a3c0ba47ddbd219a11d9eae598) 81

[PWINSTATIONCONFIG](#section_379fe706cf87417eb1f2e8536013132d) 67

[PWINSTATIONCONFIG2](#section_38d998ad51e744358231fff55348fff8) 67

[PWINSTATIONCREATE](#section_dcd01340508440898173ad3a5831e087) 67

[PWINSTATIONINFORMATION](#section_c2566d8b7016440bb7e00d07c3b2418f) 54

[PWINSTATIONLOADINDICATORDATA](#section_91459fa177e84987a6f0fe7dd3e62bfc) 74

[PWINSTATIONPRODID](#section_42e13db86eb349d3a66762500e568db5) 76

[PWINSTATIONPRODUCTINFO](#section_a75d3c4cd8c744a38a9327274236437e) 86

[PWINSTATIONREMOTEADDRESS](#section_5b5a44b3155f4159aad9794e97275a3f) 77

[PWINSTATIONSHADOW](#section_5ffd248b69324937b6044ca4456e8028) 75

[PWINSTATIONUSERTOKEN](#section_07f9831c633143e5ba27d3d58772eb4c) 74

[PWINSTATIONVALIDATIONINFORMATION](#section_5c439d8db58b4b0da2e5467aa05f993b) 85

[PWINSTATIONVIDEODATA](#section_5f95f65789d2472db4abb0595618dbd1) 74

Q

[QUERY\_SESSION\_DATA\_TYPE enumeration](#section_8244f1b1ce6246f68b62949abd789b0f) 27

R

[RCM\_REMOTEADDRESS structure](#section_3c5dea7cf9a441c6b9f831b5bc985b6b) 81

[RCMListener method](#section_40742e8650ac4bb696a619e629a06aff) 128

[RCMPublic method](#section_3cfda13d57b64161a75aa654c9bc7428) 121

[RECEIVEFLOWCONTROLCLASS enumeration](#section_611c7f925f464bd3ad49c59b9072882c) 61

[RECONNECT\_TYPE enumeration](#section_f368f3a624824943b034bd2753de12e7) 23

[References](#section_5a1c340db3d242de9d27edf03249798f) 12

[informative](#section_2ccce14446044fa8ac10695d91e459f4) 14

[normative](#section_202a2f5f8297429483efb7b8e92fa3f1) 12

[Relationship to other protocols](#section_39217b7690594ab89bc2b55bbea34ef9) 15

[RESERVED\_CACHE structure](#section_9c0c0474ac364315b0ba8fca36b5d6e5) 58

[RpcCloseEnum method](#section_cb9386d5d99e46ff8199563f6b59ecb8) 113

[RpcCloseListener method](#section_95be2185992148b0b722848ae3075b9d) 129

[RpcCloseSession method](#section_8b79657308b1450e840379ca8fdd80c7) 100

[RpcConnect method](#section_b57917b9b00648b3ae64016771c25bee) 100

[RpcConnectCallback method](#section_93794f61e93046f0a2a63f84531546ea) 172

[RpcDisconnect method](#section_dd882322dcd44c3394e5a1f8e8354ebd) 101

[RpcEnumAddFilter method](#section_2cd006cdfb1043a6a16db01379f9cb53) 114

[RpcFilterByCallersName method](#section_7672f2e67b124dc290b0ad43a231d83e) 114

[RpcFilterBySessionType method](#section_fd4db5fa13b3490b95283cb108544a4e) 115

[RpcFilterByState method](#section_14ed1d6871f2449599460116f87ee1ee) 113

[RpcGetAllListeners method](#section_4655b71d39b747258e78175cb5608195) 125

[RpcGetAllSessions method](#section_68e824e0d78649878ccbc34503fffd3f) 117

[RpcGetAllSessionsEx method](#section_3f72497925cd41d89829244f4425f9c6) 118

[RpcGetClientData method](#section_7f7d695c97f84b219ebf4580a8dab066) 122

[RpcGetConfigData method](#section_6c4dc21eea534785b50593d23604f2c8) 123

[RpcGetEnumResult method](#section_1a7d5d1d1ce5448fbb4ac79741982edb) 114

[RpcGetEnumResultEx method](#section_cb2be08441dd44ec81dc52d371a28554) 116

[RpcGetLastInputTime method](#section_fd64f3b50b1e4ed6a1fa81d7fea553fd) 124

[RpcGetLoggedOnCount method](#section_8e02daa60b31494aa3c48c8d73faf442) 107

[RpcGetProtocolStatus method](#section_954007d7e5ec483a82afd66ac8c647df) 123

[RpcGetRemoteAddress method](#section_4ee5f612b74c4e47811ce57edaf02c5b) 124

[RpcGetSessionCounters method](#section_eeda3f5626de4a6da0737ec5390d7dbd) 106

[RpcGetSessionIds method](#section_922264ed0a2d441fa71c2d3ffa975814) 116

[RpcGetSessionInformation method](#section_1a7fa234480540d3a41c8b55bde2eda0) 106

[RpcGetSessionInformationEx method](#section_7a6b39aa61a34a89b486520e81f4f9e1) 108

[RpcGetSessionIP method](#section_bf669418ceeb4df399c946bc70d3942a) 177

[RpcGetSessionProtocolLastInputTime method](#section_03f9b3dc041447f496da5a49e6c1e635) 125

[RpcGetSessionType method](#section_e7deb145e33f4ec0a2ac524b0381493d) 107

[RpcGetState method](#section_58b06e1dbfe44a2f90a19c9129923e99) 103

[RpcGetTerminalName method](#section_e98ee550643046808ac34bbc25a91a93) 102

[RpcGetTimes method](#section_4a7d8496a6c44162adaf7897baa8c4ff) 105

[RpcGetUserCertificates method](#section_aa5f03ee828448f0a3e03c4661336541) 126

[RpcGetUserName method](#section_acc0609748ad41a2a0fbfc03e4eba7c5) 102

[RpcIcaServerPing method](#section_fb4fa20ad77548b6b9b6edab02296225) 140

[RpcIsListening method](#section_13bb2b29a4b048e0a1d2d73ea67b3069) 130

[RpcIsSessionDesktopLocked method](#section_0a12b501a52c41e682b306638cc0ae63) 103

[RpcLogoff method](#section_6dbdc16149c744deb6aa80283ce07e37) 102

[RpcLogonIdFromWinStationName method](#section_b1397fdad4f24355a936ca44ef4dc440) 151

[RpcOpenEnum method](#section_f17f4b8d8d524538acf3cdfbe581a99b) 112

[RpcOpenListener method](#section_126bc7fa9e644b7ba78c1a08895fb60a) 128

[RpcOpenSession method](#section_56ede88b913b49119540389fddf879e5) 99

[RpcQuerySessionData method](#section_8661c3aabc4741b996625762ee052c2f) 127

[RpcRegisterAsyncNotification method](#section_bb3d1c58f397472e82ed44c242846464) 110

[RpcShadow2 method](#section_70cb89cf10d0429e8f8bc2c4eea5eb8c) 178

[RpcShowMessageBox method](#section_63915a11630d4ab686d11034fde1e386) 104

[RpcStartListener method](#section_e3aa6981665a4001b5bc996ba68c3f0e) 130

[RpcStopListener method](#section_64f7fe27fea14e318495f07aae42aed1) 129

[RpcUnRegisterAsyncNotification method](#section_886923e61d674ad2a79ac24c9398b224) 111

[RpcWaitAsyncNotification method](#section_81d5ba342ae446149aca8deb1eb7a0a1) 110

[RpcWaitForSessionState method](#section_637ac10c6570460380223c34ea2ea73a) 109

[RpcWinStationBreakPoint method](#section_5b55d7a032c5494ba1a91d25d283c542) 159

[RpcWinStationCheckLoopBack method](#section_e8cb245a90a3424db9bdacec6e01d86a) 171

[RpcWinStationCloseServer method](#section_89048acf28c64d469742eef51f507f05) 140

[RpcWinStationCloseServerEx method](#section_9d198d98b7f1475d8c29cc188f5bc0fc) 169

[RpcWinStationConnect method](#section_88a127d180de407aa95b9fb9d45e29d4) 152

[RpcWinStationDisconnect method](#section_194a4899474546ae9697b44a5457d77d) 153

[RpcWinStationEnumerate method](#section_fa654a9ca67f4e589fa75dd0ae6305a1) 141

[RpcWinStationEnumerateProcesses method](#section_3559bc198ca34bcf842ecf25f592dc1a) 161

[RpcWinStationGetAllProcesses method](#section_6f43f9e42d804c5cbc0b7f653b1d7c02) 163

[RpcWinStationGetAllProcesses\_NT6 method](#section_e1647da01159428491bc2c2024b14e20) 173

[RpcWinStationGetLanAdapterName method](#section_f3450fddd5b54a3190893f60b6d82306) 166

[RpcWinStationGetMachinePolicy method](#section_cda4de46ea1c49aebeb0afd0f0be90f4) 171

[RpcWinStationGetProcessSid method](#section_94fc771f75834a4d8579efac05820551) 164

[RpcWinStationGetTermSrvCountersValue method](#section_46f107c4f0904802b6d8c510cfcb9a93) 165

[RpcWinStationIsHelpAssistantSession method](#section_113108ba2939435ea14fd45414b8ad91) 170

[RpcWinStationNameFromLogonId method](#section_d2c62b1094894e2b9dd0efd6d7850881) 152

[RpcWinStationOpenServer method](#section_fd6f339df5dd452ca9a9ec753c22a36a) 139

[RpcWinStationOpenSessionDirectory method](#section_8bd6683781b74da0809ef797bd3a5b41) 174

[RpcWinStationQueryInformation method](#section_1bba9ff271d349a3bb262e5f6fcab3ee) 143

[RpcWinStationReadRegistry method](#section_5d74a6b079fd4091a2eb4974131b4ea4) 160

[RpcWinStationReInitializeSecurity method](#section_563daea7578644f3bf332d24e058856a) 166

[RpcWinStationRename method](#section_84c75092c96a4beeac508f1632bc2f7b) 142

[RpcWinStationReset method](#section_2777bdbaa4c7480bae34fb234d207bd2) 154

[RpcWinStationSendMessage method](#section_a8f4bc6097484bcab231df84330b0754) 149

[RpcWinStationSetInformation method](#section_2a5ee131a1dd44c7988098df708061ea) 147

[RpcWinStationShadow method](#section_8115399df159477fa871522fde0cc32a) 158

[RpcWinStationShadowStop method](#section_23f97d498b344ee68af087114260e2bb) 168

[RpcWinStationShutdownSystem method](#section_81c472be2a71424e8bde71e826a34c26) 155

[RpcWinStationTerminateProcess method](#section_60d456d751d9432199b0e22852573810) 162

[RpcWinStationUpdateSettings method](#section_ff68952dedfe4f81aaf2b2e37d2b29ef) 167

[RpcWinStationWaitSystemEvent method](#section_e901b3d9eb9a43af8c929feacdf9cbc6) 156

S

[Schema elements - directory service](#section_9b65ec6ba9c7401fbc4a0f20538004a1) 86

[SDCLASS enumeration](#section_7de560c9ca6b4741a448c67de0e52e0e) 22

Security

[implementer considerations](#section_81b1b9b4338040b9ab06bb1b20cbf2e4) 191

[parameter index](#section_3ecc8b1f5f7f46f48c47ddd7594a4f70) 191

Sequencing rules

[client](#section_42c14d8f56a64444a194e9586f145c56) 131

[legacy](#section_42c14d8f56a64444a194e9586f145c56) 131

[local session manager](#section_3f92d2e1f9574e7580f3ea700851535c) 95

[TermSrv](#section_04c44f66db064ac28c2b9a7bc2e9f2af) 119

[virtual IP](#section_26a3d2a761224b7d99767e7200e4d447) 176

server ([section 3.5.4](#section_24e967207f93438e996e724b2d56ce4c) 121, [section 3.7.4](#section_2268c9823653425c86628f038a8f4f2f) 134)

[legacy](#section_2268c9823653425c86628f038a8f4f2f) 134

[local session manager](#section_103be5dd283b4a41bf030430484ff583) 98

[TermSrv](#section_24e967207f93438e996e724b2d56ce4c) 121

[virtual IP](#section_0089cf899f8345f081036c261c8ef302) 176

Server

abstract data model ([section 3.3.1](#section_fc7b89539126492f95ea1554b2023fce) 95, [section 3.5.1](#section_b164fe5545f045da852bc79fffc1954b) 119, [section 3.7.1](#section_29c1c1433b824d4a880aa01f0274762c) 131)

initialization ([section 3.3.3](#section_de25f861bfb94f68aea51026292bbfb3) 98, [section 3.5.3](#section_f1ba74cb7e5340fba255e43fbc00def0) 121, [section 3.7.3](#section_caa1aed79f37455781da8e44fb5d3382) 134)

legacy

[abstract data model](#section_29c1c1433b824d4a880aa01f0274762c) 131

[initialization](#section_caa1aed79f37455781da8e44fb5d3382) 134

[local events](#section_b30c369ccba64f20a6ae4df50a021211) 175

[message processing](#section_2268c9823653425c86628f038a8f4f2f) 134

[sequencing rules](#section_2268c9823653425c86628f038a8f4f2f) 134

[timer events](#section_34d19b4fb0e84dddab178ede7c20a310) 175

[timers](#section_550cbf95ef0f446b8bddff21af420aa2) 134

[LegacyApi method](#section_d9f6ec940483417695325b47455133ed) 134

local events ([section 3.3.6](#section_06a8eb514215444da30185cd71adf259) 119, [section 3.5.6](#section_94365f4f376c4bdb876b8cfe80bd0921) 131, [section 3.7.6](#section_b30c369ccba64f20a6ae4df50a021211) 175)

local session manager

[abstract data model](#section_fc7b89539126492f95ea1554b2023fce) 95

[initialization](#section_de25f861bfb94f68aea51026292bbfb3) 98

[local events](#section_06a8eb514215444da30185cd71adf259) 119

[message processing](#section_103be5dd283b4a41bf030430484ff583) 98

[sequencing rules](#section_103be5dd283b4a41bf030430484ff583) 98

[timer events](#section_dede264ad2dc460d9681f32d4a013885) 119

[timers](#section_47fb618362474c94adb598eb067771d7) 98

message processing ([section 3.5.4](#section_24e967207f93438e996e724b2d56ce4c) 121, [section 3.7.4](#section_2268c9823653425c86628f038a8f4f2f) 134)

[RCMListener method](#section_40742e8650ac4bb696a619e629a06aff) 128

[RCMPublic method](#section_3cfda13d57b64161a75aa654c9bc7428) 121

sequencing rules ([section 3.5.4](#section_24e967207f93438e996e724b2d56ce4c) 121, [section 3.7.4](#section_2268c9823653425c86628f038a8f4f2f) 134)

TermSrv

[abstract data model](#section_b164fe5545f045da852bc79fffc1954b) 119

[initialization](#section_f1ba74cb7e5340fba255e43fbc00def0) 121

[local events](#section_94365f4f376c4bdb876b8cfe80bd0921) 131

[message processing](#section_24e967207f93438e996e724b2d56ce4c) 121

[sequencing rules](#section_24e967207f93438e996e724b2d56ce4c) 121

[timer events](#section_79450165a34d4bdb93986e33b20c7970) 130

[timers](#section_ad415e66c86b41d99572f79b4e1c0dd7) 121

timer events ([section 3.3.5](#section_dede264ad2dc460d9681f32d4a013885) 119, [section 3.5.5](#section_79450165a34d4bdb93986e33b20c7970) 130, [section 3.7.5](#section_34d19b4fb0e84dddab178ede7c20a310) 175)

timers ([section 3.3.2](#section_47fb618362474c94adb598eb067771d7) 98, [section 3.5.2](#section_ad415e66c86b41d99572f79b4e1c0dd7) 121, [section 3.7.2](#section_550cbf95ef0f446b8bddff21af420aa2) 134)

virtual IP

[abstract data model](#section_fe9fa64a3df14280ae3e1ac4812eae3a) 176

[initialization](#section_7443fec9e2094eb8a4152079cd0deab9) 176

[local events](#section_20297b8e7b8940b8abfefaee7fba8b07) 177

[message processing](#section_0089cf899f8345f081036c261c8ef302) 176

[sequencing rules](#section_0089cf899f8345f081036c261c8ef302) 176

[timer events](#section_a290eb1fec724285944091a5af84e42a) 177

[timers](#section_3827648303a146af86739663944e2a67) 176

[SESSION\_CHANGE structure](#section_e22785c95a374a47864aed2865d053bc) 81

[SESSION\_FILTER enumeration](#section_352695847c5e441081951aa49559274e) 27

[SESSIONENUM structure](#section_fd8759ef4f6c45c6b5429d5e53e90f7d) 28

[SESSIONENUM\_EX structure](#section_35a722eb00a04ba08e4c735d06378675) 30

[SESSIONENUM\_LEVEL1 structure](#section_0c4fde3b4fdb45248cae792d0320247a) 29

[SESSIONENUM\_LEVEL2 structure](#section_1b889bfe8d324cbabd92f9d7c09d5f60) 29

[SESSIONENUM\_LEVEL3 structure](#section_d1662f979e3a4fbbbdd52050df69e902) 29

[SESSIONID structure](#section_e5c755e86d66494d873f4540267fe2f4) 37

[SESSIONTYPE enumeration](#section_12510151f01d4984a1bb64ec4298c31c) 25

[SHADOW\_CONTROL\_REQUEST enumeration](#section_7229bedb8579462da2c1b07df509ebcc) 25

[SHADOW\_PERMISSION\_REQUEST enumeration](#section_71f2bab8875847719b9269aa5e6de8c5) 26

[SHADOW\_REQUEST\_RESPONSE enumeration](#section_c1534097ced840cc8d396e43c0361d33) 26

[SHADOWCLASS enumeration](#section_4099e9b221ae4a798798c627107d028f) 22

[SHADOWSTATECLASS enumeration](#section_c55fbd8fd7e34efe9ca6d0985dab9602) 76

[Standards assignments](#section_4f5ff56828874c508b3638bb85170b58) 16

[Structures](#section_5dbf8683a41343988dd34e0122d2443b) 27

[SYSTEM\_THREAD\_INFORMATION structure](#section_e82d73e4cedb40779099d58f3459722f) 42

T

[TermService listener example](#section_d045ce70be964d2392a45b842893b332) 182

[Termservice listener example example](#section_d045ce70be964d2392a45b842893b332) 182

TermSrv client

[abstract data model](#section_2c58ef6bd66544fc952df7bda2f73521) 119

[initialization](#section_880c60d039a24111a8f645df85df405d) 119

[local events](#section_b8cea0b6d52747f39a8d3ca470a9385a) 119

[message processing](#section_04c44f66db064ac28c2b9a7bc2e9f2af) 119

[sequencing rules](#section_04c44f66db064ac28c2b9a7bc2e9f2af) 119

[timer events](#section_7b2bf211c7a74f79b6967f391e3eb1e4) 119

[timers](#section_7128dcf9ea4d46aab671822f35d4deab) 119

TermSrv server

[abstract data model](#section_b164fe5545f045da852bc79fffc1954b) 119

[initialization](#section_f1ba74cb7e5340fba255e43fbc00def0) 121

[local events](#section_94365f4f376c4bdb876b8cfe80bd0921) 131

[message processing](#section_24e967207f93438e996e724b2d56ce4c) 121

[sequencing rules](#section_24e967207f93438e996e724b2d56ce4c) 121

[timer events](#section_79450165a34d4bdb93986e33b20c7970) 130

[timers](#section_ad415e66c86b41d99572f79b4e1c0dd7) 121

[TermSrvBindSecure example](#section_bc8c6536baff466bb12e435b66577bbd) 184

[Termsrvbindsecure example example](#section_bc8c6536baff466bb12e435b66577bbd) 184

[THINWIRECACHE structure](#section_3adaf1a0f6324440aeca103d9e4097ea) 58

Timer events

client ([section 3.2.5](#section_7b909a4657b14fa7adfc119334458beb) 95, [section 3.4.5](#section_7b2bf211c7a74f79b6967f391e3eb1e4) 119, [section 3.6.5](#section_566e82d44ea74a37ad805774304ba69c) 131)

[legacy](#section_566e82d44ea74a37ad805774304ba69c) 131

[local session manager](#section_7b909a4657b14fa7adfc119334458beb) 95

[TermSrv](#section_7b2bf211c7a74f79b6967f391e3eb1e4) 119

[virtual IP](#section_39b146adc1544e5ebab07b4c97b712e8) 176

server ([section 3.3.5](#section_dede264ad2dc460d9681f32d4a013885) 119, [section 3.5.5](#section_79450165a34d4bdb93986e33b20c7970) 130, [section 3.7.5](#section_34d19b4fb0e84dddab178ede7c20a310) 175)

[legacy](#section_34d19b4fb0e84dddab178ede7c20a310) 175

[local session manager](#section_dede264ad2dc460d9681f32d4a013885) 119

[TermSrv](#section_79450165a34d4bdb93986e33b20c7970) 130

[virtual IP](#section_a290eb1fec724285944091a5af84e42a) 177

Timers

client ([section 3.2.2](#section_f20440c6f3cd4e9786f443cfa8cb5044) 95, [section 3.4.2](#section_7128dcf9ea4d46aab671822f35d4deab) 119, [section 3.6.2](#section_215729887a554f9890fb9088bdb68108) 131)

[legacy](#section_215729887a554f9890fb9088bdb68108) 131

[local session manager](#section_f20440c6f3cd4e9786f443cfa8cb5044) 95

[TermSrv](#section_7128dcf9ea4d46aab671822f35d4deab) 119

[virtual IP](#section_49bf96df07a1467eb9496ed42658e987) 175

server ([section 3.3.2](#section_47fb618362474c94adb598eb067771d7) 98, [section 3.5.2](#section_ad415e66c86b41d99572f79b4e1c0dd7) 121, [section 3.7.2](#section_550cbf95ef0f446b8bddff21af420aa2) 134)

[legacy](#section_550cbf95ef0f446b8bddff21af420aa2) 134

[local session manager](#section_47fb618362474c94adb598eb067771d7) 98

[TermSrv](#section_ad415e66c86b41d99572f79b4e1c0dd7) 121

[virtual IP](#section_3827648303a146af86739663944e2a67) 176

[Tracking changes](#section_c1079a6754f348e58b57765a499670ce) 243

[TRANSMITFLOWCONTROLCLASS enumeration](#section_dd8db2b56d754adb807500d03caa0749) 61

[Transport](#section_ac0535d0a8474386ae3ee93a50ff2484) 17

[TS\_ALL\_PROCESSES\_INFO structure](#section_8f5564a61b5e4381afc44acf7ad39d6d) 38

[TS\_ALL\_PROCESSES\_INFO\_NT6 structure](#section_d60fc5882cc744649c53879142f5a58d) 40

[TS\_COUNTER structure](#section_a0173488c4834340ab1aaab2d21ea323) 43

[TS\_COUNTER\_HEADER structure](#section_e08317cba69340d39a5f61e14cfdb431) 44

[TS\_PROCESS\_INFORMATION\_NT4 structure](#section_0a8a73d93c9e4d2d91e47fef48b6c29d) 38

[TS\_SYS\_PROCESS\_INFORMATION structure](#section_96702c7d6f5f49658f295fdbea73278e) 38

[TS\_SYS\_PROCESS\_INFORMATION\_NT6 structure](#section_66ef46a7504f469696130bd8446ee225) 41

[TS\_SYSTEMTIME structure](#section_9da70cb2736e41f799e1376ed07f9f7b) 53

[TS\_TIME\_ZONE\_INFORMATION structure](#section_5e4e1129e80b4631954bbe5cfbea811f) 52

[TS\_TRACE structure](#section_ad34957553694830a1741920a7af8d5f) 78

[TS\_UNICODE\_STRING structure](#section_c90753f2f9f9490d846d6bdd41eae7f8) 40

[TSHARE\_CACHE structure](#section_07167f99bd894338b3c77aafcc75334f) 58

[TSHARE\_COUNTERS structure](#section_9981de84885c4ff8a8709912a8a5ff12) 57

[TSProperty packet](#section_0bd1a91d3cd64e97b5fa5cf35d90f005) 88

[TSVIP\_SOCKADDR structure](#section_53383eb1f61f401389905a5c7b95e10b) 83

[TSVIPAddress structure](#section_eb4b65abdbfe481db85f7931ee38c2cc) 84

[TSVIPSession structure](#section_92043576fad1495c9a5df90185959e5f) 85

U

[USERCONFIG structure](#section_dba750b8cb354e889811e2a1f8a10701) 45

[UserParameters packet](#section_15f8b01d900d4f89a3fa1681e55f383d) 87

V

[VARDATA\_WIRE structure](#section_cc15eeb5f98148d193f6d4f9bfdb07ca) 82

[Vendor-extensible fields](#section_0d8eb7e164594729bc72320557ddff32) 16

[Versioning](#section_3a25197a73e04ffb97619ab8971cff26) 16

Virtual IP client

[abstract data model](#section_c4f9189db9e54aa6b8c4f24850380ad1) 175

[initialization](#section_d8e5648f3b9a428a91d3b6abd5603152) 175

[local events](#section_4865f6f4547740859fedba1174e043f6) 176

[message processing](#section_26a3d2a761224b7d99767e7200e4d447) 176

[sequencing rules](#section_26a3d2a761224b7d99767e7200e4d447) 176

[timer events](#section_39b146adc1544e5ebab07b4c97b712e8) 176

[timers](#section_49bf96df07a1467eb9496ed42658e987) 175

Virtual IP server

[abstract data model](#section_fe9fa64a3df14280ae3e1ac4812eae3a) 176

[initialization](#section_7443fec9e2094eb8a4152079cd0deab9) 176

[local events](#section_20297b8e7b8940b8abfefaee7fba8b07) 177

[message processing](#section_0089cf899f8345f081036c261c8ef302) 176

[sequencing rules](#section_0089cf899f8345f081036c261c8ef302) 176

[timer events](#section_a290eb1fec724285944091a5af84e42a) 177

[timers](#section_3827648303a146af86739663944e2a67) 176

W

[WDCONFIG structure](#section_d1bf099beb544ed9a7230b1062dbc128) 65

[WINSTACONFIGWIRE structure](#section_ae5355e7c6cc48abb82ad1de8b3cd6d6) 83

[WINSTATIONCLIENT structure](#section_673d8ac0f55748cb98a649925160d729) 49

[WINSTATIONCLIENTDATA structure](#section_d96b68a3c0ba47ddbd219a11d9eae598) 81

[WINSTATIONCONFIG structure](#section_379fe706cf87417eb1f2e8536013132d) 67

[WINSTATIONCONFIG2 structure](#section_38d998ad51e744358231fff55348fff8) 67

[WINSTATIONCREATE structure](#section_dcd01340508440898173ad3a5831e087) 67

[WINSTATIONINFOCLASS enumeration](#section_f333c223de8a46e1a83e79cbdab92371) 19

[WINSTATIONINFORMATION structure](#section_c2566d8b7016440bb7e00d07c3b2418f) 54

[WINSTATIONLOADINDICATORDATA structure](#section_91459fa177e84987a6f0fe7dd3e62bfc) 74

[WINSTATIONPRODID structure](#section_42e13db86eb349d3a66762500e568db5) 76

[WINSTATIONPRODUCTINFO structure](#section_a75d3c4cd8c744a38a9327274236437e) 86

[WINSTATIONREMOTEADDRESS structure](#section_5b5a44b3155f4159aad9794e97275a3f) 77

[WINSTATIONSHADOW structure](#section_5ffd248b69324937b6044ca4456e8028) 75

[WINSTATIONSTATECLASS enumeration](#section_b619d8aa4e8a4716954ea4d596ed37f8) 21

[WINSTATIONUSERTOKEN structure](#section_07f9831c633143e5ba27d3d58772eb4c) 74

[WINSTATIONVALIDATIONINFORMATION structure](#section_5c439d8db58b4b0da2e5467aa05f993b) 85

[WINSTATIONVIDEODATA structure](#section_5f95f65789d2472db4abb0595618dbd1) 74