**[MS-FASP]:**

**Firewall and Advanced Security 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 |
| --- | --- | --- | --- |
| 4/3/2007 | 0.01 | New | Version 0.01 release |
| 7/3/2007 | 1.0 | Major | MLonghorn+90 |
| 7/20/2007 | 1.0.1 | Editorial | Changed language and formatting in the technical content. |
| 8/10/2007 | 1.0.2 | Editorial | Changed language and formatting in the technical content. |
| 9/28/2007 | 1.0.3 | Editorial | Changed language and formatting in the technical content. |
| 10/23/2007 | 1.0.4 | Editorial | Changed language and formatting in the technical content. |
| 11/30/2007 | 1.1 | Minor | Clarified the meaning of the technical content. |
| 1/25/2008 | 1.1.1 | Editorial | Changed language and formatting in the technical content. |
| 3/14/2008 | 1.2 | Minor | Clarified the meaning of the technical content. |
| 5/16/2008 | 2.0 | Major | Updated and revised the technical content. |
| 6/20/2008 | 2.1 | Minor | Clarified the meaning of the technical content. |
| 7/25/2008 | 3.0 | Major | Updated and revised the technical content. |
| 8/29/2008 | 4.0 | Major | Updated and revised the technical content. |
| 10/24/2008 | 4.0.1 | Editorial | Changed language and formatting in the technical content. |
| 12/5/2008 | 5.0 | Major | Updated and revised the technical content. |
| 1/16/2009 | 6.0 | Major | Updated and revised the technical content. |
| 2/27/2009 | 7.0 | Major | Updated and revised the technical content. |
| 4/10/2009 | 7.0.1 | Editorial | Changed language and formatting in the technical content. |
| 5/22/2009 | 8.0 | Major | Updated and revised the technical content. |
| 7/2/2009 | 8.0.1 | Editorial | Changed language and formatting in the technical content. |
| 8/14/2009 | 8.1 | Minor | Clarified the meaning of the technical content. |
| 9/25/2009 | 8.2 | Minor | Clarified the meaning of the technical content. |
| 11/6/2009 | 9.0 | Major | Updated and revised the technical content. |
| 12/18/2009 | 9.0.1 | Editorial | Changed language and formatting in the technical content. |
| 1/29/2010 | 9.1 | Minor | Clarified the meaning of the technical content. |
| 3/12/2010 | 9.2 | 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 | 11.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 8/27/2010 | 11.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 10/8/2010 | 11.1 | Minor | Clarified the meaning of the technical content. |
| 11/19/2010 | 11.2 | Minor | Clarified the meaning of the technical content. |
| 1/7/2011 | 11.3 | Minor | Clarified the meaning of the technical content. |
| 2/11/2011 | 12.0 | Major | Updated and revised the technical content. |
| 3/25/2011 | 13.0 | Major | Updated and revised the technical content. |
| 5/6/2011 | 14.0 | Major | Updated and revised the technical content. |
| 6/17/2011 | 14.1 | Minor | Clarified the meaning of the technical content. |
| 9/23/2011 | 15.0 | Major | Updated and revised the technical content. |
| 12/16/2011 | 16.0 | Major | Updated and revised the technical content. |
| 3/30/2012 | 17.0 | Major | Updated and revised the technical content. |
| 7/12/2012 | 18.0 | Major | Updated and revised 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 | 19.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 2/13/2014 | 20.0 | Major | Updated and revised the technical content. |
| 5/15/2014 | 21.0 | Major | Updated and revised the technical content. |
| 6/30/2015 | 22.0 | Major | Significantly changed the technical content. |
| 10/16/2015 | 22.1 | Minor | Clarified the meaning of the technical content. |
| 7/14/2016 | 23.0 | Major | Significantly changed the technical content. |
| 6/1/2017 | 24.0 | Major | Significantly changed 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 9](#_Toc499828751)

[1.1 Glossary 9](#_Toc499828752)

[1.2 References 12](#_Toc499828753)

[1.2.1 Normative References 12](#_Toc499828754)

[1.2.2 Informative References 12](#_Toc499828755)

[1.3 Overview 13](#_Toc499828756)

[1.4 Relationship to Other Protocols 15](#_Toc499828757)

[1.5 Prerequisites/Preconditions 16](#_Toc499828758)

[1.6 Applicability Statement 16](#_Toc499828759)

[1.7 Versioning and Capability Negotiation 16](#_Toc499828760)

[1.8 Vendor-Extensible Fields 17](#_Toc499828761)

[1.9 Standards Assignments 17](#_Toc499828762)

[2 Messages 18](#_Toc499828763)

[2.1 Transport 18](#_Toc499828764)

[2.2 Common Data Types 18](#_Toc499828765)

[2.2.1 FW\_STORE\_TYPE 18](#_Toc499828766)

[2.2.2 FW\_PROFILE\_TYPE 19](#_Toc499828767)

[2.2.3 FW\_POLICY\_ACCESS\_RIGHT 20](#_Toc499828768)

[2.2.4 FW\_IPV4\_SUBNET 21](#_Toc499828769)

[2.2.5 FW\_IPV4\_SUBNET\_LIST 21](#_Toc499828770)

[2.2.6 FW\_IPV6\_SUBNET 21](#_Toc499828771)

[2.2.7 FW\_IPV6\_SUBNET\_LIST 21](#_Toc499828772)

[2.2.8 FW\_IPV4\_ADDRESS\_RANGE 22](#_Toc499828773)

[2.2.9 FW\_IPV4\_RANGE\_LIST 22](#_Toc499828774)

[2.2.10 FW\_IPV6\_ADDRESS\_RANGE 22](#_Toc499828775)

[2.2.11 FW\_IPV6\_RANGE\_LIST 23](#_Toc499828776)

[2.2.12 FW\_PORT\_RANGE 23](#_Toc499828777)

[2.2.13 FW\_PORT\_RANGE\_LIST 23](#_Toc499828778)

[2.2.14 FW\_PORT\_KEYWORD 24](#_Toc499828779)

[2.2.15 FW\_PORTS 25](#_Toc499828780)

[2.2.16 FW\_ICMP\_TYPE\_CODE 25](#_Toc499828781)

[2.2.17 FW\_ICMP\_TYPE\_CODE\_LIST 25](#_Toc499828782)

[2.2.18 FW\_INTERFACE\_LUIDS 26](#_Toc499828783)

[2.2.19 FW\_DIRECTION 26](#_Toc499828784)

[2.2.20 FW\_INTERFACE\_TYPE 26](#_Toc499828785)

[2.2.21 FW\_ADDRESS\_KEYWORD 27](#_Toc499828786)

[2.2.22 FW\_ADDRESSES 28](#_Toc499828787)

[2.2.23 FW\_RULE\_STATUS 29](#_Toc499828788)

[2.2.24 FW\_RULE\_STATUS\_CLASS 42](#_Toc499828789)

[2.2.25 FW\_OBJECT\_CTRL\_FLAG 43](#_Toc499828790)

[2.2.26 FW\_ENFORCEMENT\_STATE 43](#_Toc499828791)

[2.2.27 FW\_OBJECT\_METADATA 45](#_Toc499828792)

[2.2.28 FW\_OS\_PLATFORM\_OP 45](#_Toc499828793)

[2.2.29 FW\_OS\_PLATFORM 46](#_Toc499828794)

[2.2.30 FW\_OS\_PLATFORM\_LIST 47](#_Toc499828795)

[2.2.31 FW\_RULE\_ORIGIN\_TYPE 47](#_Toc499828796)

[2.2.32 FW\_ENUM\_RULES\_FLAGS 48](#_Toc499828797)

[2.2.33 FW\_RULE\_ACTION 49](#_Toc499828798)

[2.2.34 FW\_RULE\_FLAGS 49](#_Toc499828799)

[2.2.35 FW\_RULE2\_0 51](#_Toc499828800)

[2.2.36 FW\_RULE 52](#_Toc499828801)

[2.2.37 FW\_PROFILE\_CONFIG 57](#_Toc499828802)

[2.2.38 FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_VALUES 59](#_Toc499828803)

[2.2.39 FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING\_VALUES 60](#_Toc499828804)

[2.2.40 FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_VALUES 60](#_Toc499828805)

[2.2.41 FW\_GLOBAL\_CONFIG 61](#_Toc499828806)

[2.2.42 FW\_CONFIG\_FLAGS 64](#_Toc499828807)

[2.2.43 FW\_NETWORK 64](#_Toc499828808)

[2.2.44 FW\_ADAPTER 64](#_Toc499828809)

[2.2.45 FW\_DIAG\_APP 65](#_Toc499828810)

[2.2.46 FW\_RULE\_CATEGORY 65](#_Toc499828811)

[2.2.47 FW\_PRODUCT 65](#_Toc499828812)

[2.2.48 FW\_IP\_VERSION 66](#_Toc499828813)

[2.2.49 FW\_IPSEC\_PHASE 66](#_Toc499828814)

[2.2.50 FW\_CS\_RULE\_FLAGS 67](#_Toc499828815)

[2.2.51 FW\_CS\_RULE\_ACTION 68](#_Toc499828816)

[2.2.52 FW\_CS\_RULE2\_10 68](#_Toc499828817)

[2.2.53 FW\_CS\_RULE2\_0 69](#_Toc499828818)

[2.2.54 FW\_CS\_RULE 70](#_Toc499828819)

[2.2.55 FW\_CERT\_CRITERIA\_TYPE 74](#_Toc499828820)

[2.2.56 FW\_CERT\_CRITERIA\_NAME\_TYPE 74](#_Toc499828821)

[2.2.57 FW\_CERT\_CRITERIA\_FLAGS 75](#_Toc499828822)

[2.2.58 FW\_CERT\_CRITERIA 75](#_Toc499828823)

[2.2.59 FW\_AUTH\_METHOD 76](#_Toc499828824)

[2.2.60 FW\_AUTH\_SUITE\_FLAGS 77](#_Toc499828825)

[2.2.61 FW\_AUTH\_SUITE2\_10 78](#_Toc499828826)

[2.2.62 FW\_AUTH\_SUITE 79](#_Toc499828827)

[2.2.63 FW\_AUTH\_SET2\_10 80](#_Toc499828828)

[2.2.64 FW\_AUTH\_SET 83](#_Toc499828829)

[2.2.65 FW\_CRYPTO\_KEY\_EXCHANGE\_TYPE 86](#_Toc499828830)

[2.2.66 FW\_CRYPTO\_ENCRYPTION\_TYPE 87](#_Toc499828831)

[2.2.67 FW\_CRYPTO\_HASH\_TYPE 88](#_Toc499828832)

[2.2.68 FW\_CRYPTO\_PROTOCOL\_TYPE 89](#_Toc499828833)

[2.2.69 FW\_PHASE1\_CRYPTO\_SUITE 89](#_Toc499828834)

[2.2.70 FW\_PHASE2\_CRYPTO\_SUITE 90](#_Toc499828835)

[2.2.71 FW\_PHASE1\_CRYPTO\_FLAGS 91](#_Toc499828836)

[2.2.72 FW\_PHASE2\_CRYPTO\_PFS 91](#_Toc499828837)

[2.2.73 FW\_CRYPTO\_SET 92](#_Toc499828838)

[2.2.74 FW\_BYTE\_BLOB 95](#_Toc499828839)

[2.2.75 FW\_COOKIE\_PAIR 96](#_Toc499828840)

[2.2.76 FW\_PHASE1\_KEY\_MODULE\_TYPE 96](#_Toc499828841)

[2.2.77 FW\_CERT\_INFO 96](#_Toc499828842)

[2.2.78 FW\_AUTH\_INFO 97](#_Toc499828843)

[2.2.79 FW\_ENDPOINTS 97](#_Toc499828844)

[2.2.80 FW\_PHASE1\_SA\_DETAILS 98](#_Toc499828845)

[2.2.81 FW\_PHASE2\_TRAFFIC\_TYPE 99](#_Toc499828846)

[2.2.82 FW\_PHASE2\_SA\_DETAILS 99](#_Toc499828847)

[2.2.83 FW\_PROFILE\_CONFIG\_VALUE 100](#_Toc499828848)

[2.2.84 FW\_MM\_RULE 101](#_Toc499828849)

[2.2.85 FW\_CONN\_HANDLE 103](#_Toc499828850)

[2.2.86 FW\_MATCH\_KEY 103](#_Toc499828851)

[2.2.87 FW\_DATA\_TYPE 104](#_Toc499828852)

[2.2.88 FW\_MATCH\_VALUE 105](#_Toc499828853)

[2.2.89 FW\_MATCH\_TYPE 105](#_Toc499828854)

[2.2.90 FW\_QUERY\_CONDITION 106](#_Toc499828855)

[2.2.91 FW\_QUERY\_CONDITIONS 107](#_Toc499828856)

[2.2.92 FW\_QUERY 107](#_Toc499828857)

[2.2.93 FW\_POLICY\_STORE\_HANDLE 108](#_Toc499828858)

[2.2.94 FW\_PRODUCT\_HANDLE 108](#_Toc499828859)

[2.2.95 FW\_KEY\_MODULE 108](#_Toc499828860)

[2.2.96 FW\_TRUST\_TUPLE\_KEYWORD 109](#_Toc499828861)

[2.2.97 FW\_RULE2\_10 110](#_Toc499828862)

[2.2.98 FW\_AUTH\_SET\_FLAGS 111](#_Toc499828863)

[2.2.99 FW\_CRYPTO\_SET\_FLAGS 111](#_Toc499828864)

[2.2.100 FW\_NETWORK\_NAMES 111](#_Toc499828865)

[2.2.101 FW\_RULE2\_20 112](#_Toc499828866)

[2.2.102 FW\_RULE\_FLAGS2 113](#_Toc499828867)

[2.2.103 FW\_RULE2\_24 113](#_Toc499828868)

[2.2.104 FW\_RULE2\_25 114](#_Toc499828869)

[2.2.105 FW\_RULE2\_26 115](#_Toc499828870)

[3 Protocol Details 117](#_Toc499828871)

[3.1 Server Details 117](#_Toc499828872)

[3.1.1 Abstract Data Model 117](#_Toc499828873)

[3.1.2 Timers 120](#_Toc499828874)

[3.1.3 Initialization 120](#_Toc499828875)

[3.1.4 Message Processing Events and Sequencing Rules 121](#_Toc499828876)

[3.1.4.1 RRPC\_FWOpenPolicyStore (Opnum 0) 129](#_Toc499828877)

[3.1.4.2 RRPC\_FWClosePolicyStore (Opnum 1) 130](#_Toc499828878)

[3.1.4.3 RRPC\_FWRestoreDefaults (Opnum 2) 130](#_Toc499828879)

[3.1.4.4 RRPC\_FWGetGlobalConfig (Opnum 3) 131](#_Toc499828880)

[3.1.4.5 RRPC\_FWSetGlobalConfig (Opnum 4) 132](#_Toc499828881)

[3.1.4.6 RRPC\_FWAddFirewallRule (Opnum 5) 134](#_Toc499828882)

[3.1.4.7 RRPC\_FWSetFirewallRule (Opnum 6) 135](#_Toc499828883)

[3.1.4.8 RRPC\_FWDeleteFirewallRule (Opnum 7) 136](#_Toc499828884)

[3.1.4.9 RRPC\_FWDeleteAllFirewallRules (Opnum 8) 136](#_Toc499828885)

[3.1.4.10 RRPC\_FWEnumFirewallRules (Opnum 9) 137](#_Toc499828886)

[3.1.4.11 RRPC\_FWGetConfig (Opnum 10) 138](#_Toc499828887)

[3.1.4.12 RRPC\_FWSetConfig (Opnum 11) 140](#_Toc499828888)

[3.1.4.13 RRPC\_FWAddConnectionSecurityRule (Opnum 12) 141](#_Toc499828889)

[3.1.4.14 RRPC\_FWSetConnectionSecurityRule (Opnum 13) 142](#_Toc499828890)

[3.1.4.15 RRPC\_FWDeleteConnectionSecurityRule (Opnum 14) 143](#_Toc499828891)

[3.1.4.16 RRPC\_FWDeleteAllConnectionSecurityRules (Opnum 15) 144](#_Toc499828892)

[3.1.4.17 RRPC\_FWEnumConnectionSecurityRules (Opnum 16) 145](#_Toc499828893)

[3.1.4.18 RRPC\_FWAddAuthenticationSet (Opnum 17) 146](#_Toc499828894)

[3.1.4.19 RRPC\_FWSetAuthenticationSet (Opnum 18) 147](#_Toc499828895)

[3.1.4.20 RRPC\_FWDeleteAuthenticationSet (Opnum 19) 148](#_Toc499828896)

[3.1.4.21 RRPC\_FWDeleteAllAuthenticationSets (Opnum 20) 149](#_Toc499828897)

[3.1.4.22 RRPC\_FWEnumAuthenticationSets (Opnum 21) 150](#_Toc499828898)

[3.1.4.23 RRPC\_FWAddCryptoSet (Opnum 22) 151](#_Toc499828899)

[3.1.4.24 RRPC\_FWSetCryptoSet (Opnum 23) 152](#_Toc499828900)

[3.1.4.25 RRPC\_FWDeleteCryptoSet (Opnum 24) 154](#_Toc499828901)

[3.1.4.26 RRPC\_FWDeleteAllCryptoSets (Opnum 25) 155](#_Toc499828902)

[3.1.4.27 RRPC\_FWEnumCryptoSets (Opnum 26) 156](#_Toc499828903)

[3.1.4.28 RRPC\_FWEnumPhase1SAs (Opnum 27) 157](#_Toc499828904)

[3.1.4.29 RRPC\_FWEnumPhase2SAs (Opnum 28) 158](#_Toc499828905)

[3.1.4.30 RRPC\_FWDeletePhase1SAs (Opnum 29) 159](#_Toc499828906)

[3.1.4.31 RRPC\_FWDeletePhase2SAs (Opnum 30) 160](#_Toc499828907)

[3.1.4.32 RRPC\_FWEnumProducts (Opnum 31) 161](#_Toc499828908)

[3.1.4.33 RRPC\_FWAddMainModeRule (Opnum 32) 162](#_Toc499828909)

[3.1.4.34 RRPC\_FWSetMainModeRule (Opnum 33) 162](#_Toc499828910)

[3.1.4.35 RRPC\_FWDeleteMainModeRule (Opnum 34) 163](#_Toc499828911)

[3.1.4.36 RRPC\_FWDeleteAllMainModeRules (Opnum 35) 164](#_Toc499828912)

[3.1.4.37 RRPC\_FWEnumMainModeRules (Opnum 36) 165](#_Toc499828913)

[3.1.4.38 RRPC\_FWQueryFirewallRules (Opnum 37) 166](#_Toc499828914)

[3.1.4.39 RRPC\_FWQueryConnectionSecurityRules (Opnum 38) 167](#_Toc499828915)

[3.1.4.40 RRPC\_FWQueryMainModeRules (Opnum 39) 168](#_Toc499828916)

[3.1.4.41 RRPC\_FWQueryAuthenticationSets (Opnum 40) 169](#_Toc499828917)

[3.1.4.42 RRPC\_FWQueryCryptoSets (Opnum 41) 170](#_Toc499828918)

[3.1.4.43 RRPC\_FWEnumNetworks (Opnum 42) 171](#_Toc499828919)

[3.1.4.44 RRPC\_FWEnumAdapters (Opnum 43) 171](#_Toc499828920)

[3.1.4.45 RRPC\_FWGetGlobalConfig2\_10 (Opnum 44) 172](#_Toc499828921)

[3.1.4.46 RRPC\_FWGetConfig2\_10 (Opnum 45) 174](#_Toc499828922)

[3.1.4.47 RRPC\_FWAddFirewallRule2\_10 (Opnum 46) 175](#_Toc499828923)

[3.1.4.48 RRPC\_FWSetFirewallRule2\_10 (Opnum 47) 176](#_Toc499828924)

[3.1.4.49 RRPC\_FWEnumFirewallRules2\_10 (Opnum 48) 177](#_Toc499828925)

[3.1.4.50 RRPC\_FWAddConnectionSecurityRule2\_10 (Opnum 49) 179](#_Toc499828926)

[3.1.4.51 RRPC\_FWSetConnectionSecurityRule2\_10 (Opnum 50) 180](#_Toc499828927)

[3.1.4.52 RRPC\_FWEnumConnectionSecurityRules2\_10 (Opnum 51) 181](#_Toc499828928)

[3.1.4.53 RRPC\_FWAddAuthenticationSet2\_10 (Opnum 52) 182](#_Toc499828929)

[3.1.4.54 RRPC\_FWSetAuthenticationSet2\_10 (Opnum 53) 183](#_Toc499828930)

[3.1.4.55 RRPC\_FWEnumAuthenticationSets2\_10 (Opnum 54) 184](#_Toc499828931)

[3.1.4.56 RRPC\_FWAddCryptoSet2\_10 (Opnum 55) 185](#_Toc499828932)

[3.1.4.57 RRPC\_FWSetCryptoSet2\_10 (Opnum 56) 186](#_Toc499828933)

[3.1.4.58 RRPC\_FWEnumCryptoSets2\_10 (Opnum 57) 187](#_Toc499828934)

[3.1.4.59 RRPC\_FWAddConnectionSecurityRule2\_20 (Opnum 58) 188](#_Toc499828935)

[3.1.4.60 RRPC\_FWSetConnectionSecurityRule2\_20 (Opnum 59) 189](#_Toc499828936)

[3.1.4.61 RRPC\_FWEnumConnectionSecurityRules2\_20 (Opnum 60) 190](#_Toc499828937)

[3.1.4.62 RRPC\_FWQueryConnectionSecurityRules2\_20 (Opnum 61) 191](#_Toc499828938)

[3.1.4.63 RRPC\_FWAddAuthenticationSet2\_20 (Opnum 62) 192](#_Toc499828939)

[3.1.4.64 RRPC\_FWSetAuthenticationSet2\_20 (Opnum 63) 193](#_Toc499828940)

[3.1.4.65 RRPC\_FWEnumAuthenticationSets2\_20 (Opnum 64) 194](#_Toc499828941)

[3.1.4.66 RRPC\_FWQueryAuthenticationSets2\_20 (Opnum 65) 195](#_Toc499828942)

[3.1.4.67 RRPC\_FWAddFirewallRule2\_20 (Opnum 66) 196](#_Toc499828943)

[3.1.4.68 RRPC\_FWSetFirewallRule2\_20 (Opnum 67) 197](#_Toc499828944)

[3.1.4.69 RRPC\_FWEnumFirewallRules2\_20 (Opnum 68) 198](#_Toc499828945)

[3.1.4.70 RRPC\_FWQueryFirewallRules2\_20 (Opnum 69) 199](#_Toc499828946)

[3.1.4.71 RRPC\_FWAddFirewallRule2\_24 (Opnum 70) 200](#_Toc499828947)

[3.1.4.72 RRPC\_FWSetFirewallRule2\_24 (Opnum 71) 202](#_Toc499828948)

[3.1.4.73 RRPC\_FWEnumFirewallRules2\_24 (Opnum 72) 203](#_Toc499828949)

[3.1.4.74 RRPC\_FWQueryFirewallRules2\_24 (Opnum 73) 204](#_Toc499828950)

[3.1.4.75 RRPC\_FWAddFirewallRule2\_25 (Opnum 74) 205](#_Toc499828951)

[3.1.4.76 RRPC\_FWSetFirewallRule2\_25 (Opnum 75) 206](#_Toc499828952)

[3.1.4.77 RRPC\_FWEnumFirewallRules2\_25 (Opnum 76) 207](#_Toc499828953)

[3.1.4.78 RRPC\_FWQueryFirewallRules2\_25 (Opnum 77) 208](#_Toc499828954)

[3.1.4.79 RRPC\_FWAddFirewallRule2\_26 (Opnum 78) 209](#_Toc499828955)

[3.1.4.80 RRPC\_FWSetFirewallRule2\_26 (Opnum 79) 210](#_Toc499828956)

[3.1.4.81 RRPC\_FWEnumFirewallRules2\_26 (Opnum 80) 211](#_Toc499828957)

[3.1.4.82 RRPC\_FWQueryFirewallRules2\_26 (Opnum 81) 212](#_Toc499828958)

[3.1.4.83 RRPC\_FWAddFirewallRule2\_27 (Opnum 82) 213](#_Toc499828959)

[3.1.4.84 RRPC\_FWSetFirewallRule2\_27 (Opnum 83) 214](#_Toc499828960)

[3.1.4.85 RRPC\_FWEnumFirewallRules2\_27 (Opnum 84) 215](#_Toc499828961)

[3.1.4.86 RRPC\_FWQueryFirewallRules2\_27 (Opnum 85) 216](#_Toc499828962)

[3.1.5 Timer Events 217](#_Toc499828963)

[3.1.6 Other Local Events 217](#_Toc499828964)

[3.1.6.1 AddPortInUse 217](#_Toc499828965)

[3.1.6.2 DeletePortInUse 218](#_Toc499828966)

[3.1.6.3 AddDefaultFirewallRule 218](#_Toc499828967)

[3.1.6.4 SetGroupPolicyRSoPStore 218](#_Toc499828968)

[3.1.6.5 IsComputerInCommonCriteriaMode 218](#_Toc499828969)

[3.1.6.6 SetEffectiveFirewallPolicy 219](#_Toc499828970)

[3.1.6.7 AddTrustTuple 219](#_Toc499828971)

[3.1.6.8 DeleteTrustTuple 219](#_Toc499828972)

[3.2 Client Details 219](#_Toc499828973)

[3.2.1 Abstract Data Model 219](#_Toc499828974)

[3.2.2 Timers 220](#_Toc499828975)

[3.2.3 Initialization 220](#_Toc499828976)

[3.2.4 Message Processing Events and Sequencing Rules 220](#_Toc499828977)

[3.2.5 Timer Events 220](#_Toc499828978)

[3.2.6 Other Local Events 220](#_Toc499828979)

[4 Protocol Examples 221](#_Toc499828980)

[4.1 Opening a Policy Store 221](#_Toc499828981)

[4.2 Adding a Firewall Rule 221](#_Toc499828982)

[4.3 Enumerating the Firewall Rules 223](#_Toc499828983)

[4.4 Closing a Policy Store Handle 223](#_Toc499828984)

[5 Security 225](#_Toc499828985)

[5.1 Security Considerations for Implementers 225](#_Toc499828986)

[5.2 Index of Security Parameters 225](#_Toc499828987)

[6 Appendix A: Full IDL 226](#_Toc499828988)

[7 Appendix B: Product Behavior 285](#_Toc499828989)

[8 Change Tracking 296](#_Toc499828990)

[9 Index 297](#_Toc499828991)

# Introduction

The Firewall and Advanced Security Protocol describes managing security policies on remote computers. The specific policies that this protocol manages are those of the firewall and advanced security components. The protocol allows the same functionality that is available locally; it can add, modify, delete, and enumerate policies. It can also enumerate [**security associations**](#gt_67cbf867-7a49-41f3-a68f-37b5f9035acb) that can be generated between hosts after this policy is enforced.

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

## Glossary

This document uses the following terms:

**access control entry (ACE)**: An entry in an [**access control list (ACL)**](#gt_9f92aa05-dd0a-45f2-88d6-89f1fb654395) that contains a set of user rights and a [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) that identifies a principal for whom the rights are allowed, denied, or audited.

**access control list (ACL)**: A list of [**access control entries (ACEs)**](#gt_b581857f-39aa-4979-876b-daba67a40f15) that collectively describe the security rules for authorizing access to some resource; for example, an object or set of objects.

**Authenticated IP (AuthIP)**: An [**Internet Key Exchange (IKE)**](#gt_294fef97-5790-4d41-971e-dd255b783e68) protocol extension, as specified in [[MS-AIPS]](%5bMS-AIPS%5d.pdf#Section_eee3de6438474451978e9513ff187d30).

**authentication header (AH)**: An [**Internet Protocol Security (IPsec)**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) encapsulation mode that provides authentication and message integrity. For more information, see [[RFC4302]](https://go.microsoft.com/fwlink/?LinkId=90466) section 1.

**certificate revocation list (CRL)**: A list of certificates that have been revoked by the [**certification authority (CA)**](#gt_c925d5d7-a442-4ba4-9586-5f94ccec847a) that issued them (that have not yet expired of their own accord). The list must be cryptographically signed by the [**CA**](#gt_c925d5d7-a442-4ba4-9586-5f94ccec847a) that issues it. Typically, the certificates are identified by serial number. In addition to the serial number for the revoked certificates, the CRL contains the revocation reason for each certificate and the time the certificate was revoked. As described in [[RFC3280]](https://go.microsoft.com/fwlink/?LinkId=90414), two types of CRLs commonly exist in the industry. Base CRLs keep a complete list of revoked certificates, while delta CRLs maintain only those certificates that have been revoked since the last issuance of a base CRL. For more information, see [[X509]](https://go.microsoft.com/fwlink/?LinkId=90590) section 7.3, [[MSFT-CRL]](https://go.microsoft.com/fwlink/?LinkId=90181), and [RFC3280] section 5.

**certification authority (CA)**: A third party that issues public key certificates. Certificates serve to bind public keys to a user identity. Each user and certification authority (CA) can decide whether to trust another user or CA for a specific purpose, and whether this trust should be transitive. For more information, see [RFC3280].

**common criteria mode**: A computer system is said to be operating in [**common criteria mode**](#gt_52549a11-2432-4a5c-966f-5f8a32de9162) when it conforms to all the security functional requirements specified in [[CCITSE3.1-3]](https://go.microsoft.com/fwlink/?LinkId=211804), Part 2.

**dynamic endpoint**: A network-specific server address that is requested and assigned at run time. For more information, see [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824).

**edge firewall**: A firewall that's connected to two networks: an internal network and an external network, usually the Internet.

**Encapsulating Security Payload (ESP)**: An [**Internet Protocol security (IPsec)**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) encapsulation mode that provides authentication, data confidentiality, and message integrity. For more information, see [[RFC4303]](https://go.microsoft.com/fwlink/?LinkId=90467) section 1.

**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. For more information, see [C706].

**enhanced key usage (EKU)**: An extension that is a collection of object identifiers (OIDs) that indicate the applications that use the key.

**fully qualified binary name (FQBN)**: A string constructed by the operating system that takes the format "Company\Product Suite\Product, Version" for a signed Windows binary file and that can be derived from the publishing information for such a file.

**fully qualified domain name (FQDN)**: An unambiguous domain name that gives an absolute location in the Domain Name System's (DNS) hierarchy tree, as defined in [[RFC1035]](https://go.microsoft.com/fwlink/?LinkId=90264) section 3.1 and [[RFC2181]](https://go.microsoft.com/fwlink/?LinkId=127732) section 11.

**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).

**Group Policy**: A mechanism that allows the implementer to specify managed configurations for users and computers in an Active Directory service environment.

**Group Policy Object (GPO)**: A collection of administrator-defined specifications of the policy settings that can be applied to groups of computers in a domain. Each GPO includes two elements: an object that resides in the Active Directory for the domain, and a corresponding file system subdirectory that resides on the sysvol DFS share of the Group Policy server for the domain.

**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.

**Internet Key Exchange (IKE)**: The protocol that is used to negotiate and provide authenticated keying material for [**security associations (SAs)**](#gt_67cbf867-7a49-41f3-a68f-37b5f9035acb) in a protected manner. For more information, see [[RFC2409]](https://go.microsoft.com/fwlink/?LinkId=90349).

**Internet Key Exchange (IKEv2)**: The protocol that is used to negotiate and provide authenticated keying material for security associations (SA) in a protected manner. For more information, see [[RFC4306]](https://go.microsoft.com/fwlink/?LinkId=90469).

**Internet Protocol security (IPsec)**: A framework of open standards for ensuring private, secure communications over Internet Protocol (IP) networks through the use of cryptographic security services. IPsec supports network-level peer authentication, data origin authentication, data integrity, data confidentiality (encryption), and replay protection.

**Kerberos**: An authentication system that enables two parties to exchange private information across an otherwise open network by assigning a unique key (called a ticket) to each user that logs on to the network and then embedding these tickets into messages sent by the users. For more information, see [[MS-KILE]](%5bMS-KILE%5d.pdf#Section_2a32282edd484ad9a542609804b02cc9).

**Key Distribution Center (KDC)**: The [**Kerberos**](#gt_d6a282ce-b1da-41e1-b05a-22f777a5c1fe) service that implements the authentication and ticket granting services specified in the [**Kerberos**](#gt_d6a282ce-b1da-41e1-b05a-22f777a5c1fe) protocol. The service runs on computers selected by the administrator of the realm or domain; it is not present on every machine on the network. It must have access to an account database for the realm that it serves. [**KDCs**](#gt_6e5aafba-6b66-4fdd-872e-844f142af287) are integrated into the domain controller role. It is a network service that supplies tickets to clients for use in authenticating to services.

**locally unique identifier (LUID)**: A 64-bit value guaranteed to be unique within the scope of a single machine.

**Network Data Representation (NDR)**: A specification that defines a mapping from [**Interface Definition Language (IDL)**](#gt_73177eec-4092-420f-92c5-60b2478df824) data types onto octet streams. [**NDR**](#gt_9ebf9540-2c31-43bc-bc56-4a932faabf2d) also refers to the runtime environment that implements the mapping facilities (for example, data provided to [**NDR**](#gt_9ebf9540-2c31-43bc-bc56-4a932faabf2d)). For more information, see [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) and [C706] section 14.

**perfect forward secrecy (PFS)**: A property of key exchange protocols, which holds when session keys from previous communications are not compromised by the disclosure of longer-term keying material. In the context of [**Internet Protocol security (IPsec)**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb), [**PFS**](#gt_5d8948bc-5e32-483b-906d-42f785d0df18) requires a Diffie-Hellman exchange to generate the keys for each quick mode [**security association (SA)**](#gt_67cbf867-7a49-41f3-a68f-37b5f9035acb).

**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].

**Rivest-Shamir-Adleman (RSA)**: A system for public key cryptography. [**RSA**](#gt_3f85a24a-f32a-4322-9e99-eba6ae802cd6) is specified in [[PKCS1]](https://go.microsoft.com/fwlink/?LinkId=90248) and [[RFC3447]](https://go.microsoft.com/fwlink/?LinkId=90422).

**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 association (SA)**: A simplex "connection" that provides security services to the traffic carried by it. See [[RFC4301]](https://go.microsoft.com/fwlink/?LinkId=90465) for more information.

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

**Security Support Provider Interface (SSPI)**: A Windows API that provides the means for connected applications to call one of several security providers to establish authenticated connections and to exchange data securely over those connections. It is equivalent to Generic Security Services (GSS)-API, and the two APIs are on-the-wire compatible.

**stealth mode**: A firewall is said to be operating in stealth mode when it prevents the host computer from responding to unsolicited network traffic.

**Transmission Control Protocol (TCP)**: A protocol used with the Internet Protocol (IP) to send data in the form of message units between computers over the Internet. TCP handles keeping track of the individual units of data (called packets) that a message is divided into for efficient routing through the Internet.

**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.

**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)

[CCITSE3.1-3] CCRA, "Common Criteria for Information Technology Security Evaluation", version 3.1-3, July 2009, [http://www.commoncriteriaportal.org/cc/](https://go.microsoft.com/fwlink/?LinkId=211804)

[MS-AIPS] Microsoft Corporation, "[Authenticated Internet Protocol](%5bMS-AIPS%5d.pdf#Section_eee3de6438474451978e9513ff187d30)".

[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-GPFAS] Microsoft Corporation, "[Group Policy: Firewall and Advanced Security Data Structure](%5bMS-GPFAS%5d.pdf#Section_46e8d583a4ce4c43b399566afb1eec7f)".

[MS-IKEE] Microsoft Corporation, "[Internet Key Exchange Protocol Extensions](%5bMS-IKEE%5d.pdf#Section_e05e2762179a4c3fbfb50aca7bbefe79)".

[MS-KILE] Microsoft Corporation, "[Kerberos Protocol Extensions](%5bMS-KILE%5d.pdf#Section_2a32282edd484ad9a542609804b02cc9)".

[MS-NLMP] Microsoft Corporation, "[NT LAN Manager (NTLM) Authentication Protocol](%5bMS-NLMP%5d.pdf#Section_b38c36ed28044868a9ff8dd3182128e4)".

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

[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)

[RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005, [http://www.ietf.org/rfc/rfc4306.txt](https://go.microsoft.com/fwlink/?LinkId=90469)

[X501] ITU-T, "Information Technology - Open Systems Interconnection - The Directory: The Models", Recommendation X.501, August 2005, [http://www.itu.int/rec/T-REC-X.501-200508-S/en](https://go.microsoft.com/fwlink/?LinkId=98847)

### Informative References

[IANA-PROTO-NUM] IANA, "Protocol Numbers", February 2007, [http://www.iana.org/assignments/protocol-numbers](https://go.microsoft.com/fwlink/?LinkId=89889)

[MS-DLNHND] Microsoft Corporation, "[Digital Living Network Alliance (DLNA) Networked Device Interoperability Guidelines: Microsoft Extensions](%5bMS-DLNHND%5d.pdf#Section_a12b022ab79346b2aaa96100611b5038)".

[MS-GPOL] Microsoft Corporation, "[Group Policy: Core Protocol](%5bMS-GPOL%5d.pdf#Section_62d1292462524052996f161d2b9019f4)".

[MS-GPREG] Microsoft Corporation, "[Group Policy: Registry Extension Encoding](%5bMS-GPREG%5d.pdf#Section_834da877264f45899b80b6b012c8edc3)".

[MSDN-BCryptGetFipsAlgorithmMode] Microsoft Corporation, "BCryptGetFipsAlgorithmMode function", [http://msdn.microsoft.com/en-us/library/aa375460(VS.85).aspx](https://go.microsoft.com/fwlink/?LinkId=211797)

[MSDN-ExpandEnvironmentStrings] Microsoft Corporation, "ExpandEnvironmentStrings function", [http://msdn.microsoft.com/en-us/library/ms724265(VS.85).aspx](https://go.microsoft.com/fwlink/?LinkId=211083)

[MSDN-FQBN] Microsoft Corporation, "CLAIM\_SECURITY\_ATTRIBUTE\_FQBN\_VALUE structure", [https://msdn.microsoft.com/en-us/library/system.security.claims.claimvaluetypes.fqbn(v=vs.110).aspx](https://go.microsoft.com/fwlink/?linkid=839018)

[MSDN-OSVERSIONINFOEX] Microsoft Corporation, "OSVERSIONINFOEX structure", Structure, [http://msdn.microsoft.com/en-us/library/ms724833.aspx](https://go.microsoft.com/fwlink/?LinkId=90057)

[MSDN-SHLoadIndirectString] Microsoft Corporation, "SHLoadIndirectString function", [http://msdn.microsoft.com/en-us/library/bb759919(VS.85).aspx](https://go.microsoft.com/fwlink/?LinkId=210820)

[MSKB-935807] Microsoft Corporation, "Security Update for Windows Vista (KB935807)", August 2007, [http://www.microsoft.com/downloads/details.aspx?FamilyId=e9b64746-6afa-4a30-833d-e058e000c821&displaylang=en](https://go.microsoft.com/fwlink/?LinkId=122599)

[MSWFPSDK] Microsoft Corporation, "Windows Filtering Platform", [http://msdn.microsoft.com/en-us/library/aa366510.aspx](https://go.microsoft.com/fwlink/?LinkId=90220)

[RFC2409] Harkins, D. and Carrel, D., "The Internet Key Exchange (IKE)", RFC 2409, November 1998, [http://www.ietf.org/rfc/rfc2409.txt](https://go.microsoft.com/fwlink/?LinkId=90349)

[RFC4301] Kent, S. and Seo, K., "Security Architecture for the Internet Protocol", RFC 4301, December 2005, [http://www.ietf.org/rfc/rfc4301.txt](https://go.microsoft.com/fwlink/?LinkId=90465)

## Overview

A host firewall is a software component that runs on host computers. It provides a layer of defense that can add depth to the collection of security measures when combined with other security measures, such as [**edge firewalls**](#gt_beea3068-6e40-41b8-93ae-167e6de94db3). Any threats that manage to get through the edge firewall, or those that are launched from within a corporate network, can still be defended against when host firewalls are used. Host firewalls are also useful in consumer scenarios in which there is, typically, no edge firewall to protect the home network.

[**Internet Protocol Security (IPsec)**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) is a host-based, policy-driven security solution for protecting the host from all network access. IPsec focuses on connection security, which includes authentication, integrity protection, and confidentiality (encryption) of communication.

Because both IPsec and firewalls are host-based policy security technologies that operate in the network stack, they are managed together to avoid conflicts. Furthermore, firewall and connection security (IPsec) can interact, providing deeper and more effective filtering capabilities based on identities that are negotiated by IPsec as well as other IPsec state information. This document refers to this combined security solution as the firewall and advanced security components.

Firewall and advanced security components can be governed by policy that is received from local users or from network-wide policy that is distributed by an administrator, or both. There is a need in managed environments for a network administrator to be able to monitor the policies in effect on hosts, assuming that hosts might have received policies from both sources.

Network-wide policies are usually distributed by using [**Group Policy Objects (GPOs)**](#gt_dec32233-8776-4151-91a0-8624a2b9abb0) that live on active directories of domains. However, some workgroups or networks might not have a domain infrastructure. Even in non-domain joined environments, the network administrator needs to be able to remotely manage the advanced firewall and IPsec policy of a host.

Lastly, the network administrator might also be required to diagnose problems on the remote hosts. A common technique is to create temporary changes and then see if the changes fix the problem. This is the third scenario that warrants the capability to remotely administer host policies.

The Firewall and Advanced Security Protocol is designed and used to address the three needs previously mentioned. That is, its purpose is to monitor and manage remote host policies. It can manage all the policies that an administrator can manage locally. It can also monitor the specific policies coming from the different sources or monitor them aggregated, that is, all together, to understand and predict expected behavior. Lastly, it can make temporary modifications on the remote host policy to test online fixes and see whether they are effective.

The Firewall and Advanced Security Protocol is a client/server, [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331)-based protocol. It consists of data types and methods. The data types are used to represent the different types of policy components that compose policy objects and policy configuration options. The methods are operations that are used to read and manage the different available policies. Therefore, the user can make method calls that pass new policy objects to be added to the policy, delete from the policy, or modify an existing object within the policy. The user can also call methods to retrieve all the policy objects of interest. The following illustration shows read and write operations and their message sequences.

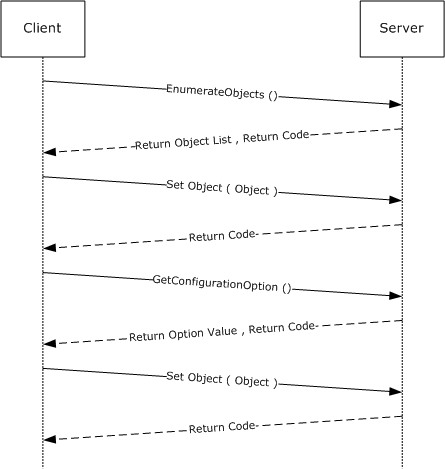


Figure 1: Read and write operations and their message sequences

The server role is represented by the host firewall, which contains the policy and enforces it. The client role is represented by the management console (or other user management tool), which sends, retrieves, and modifies the policies on the remote host firewall.

## Relationship to Other Protocols

This protocol is implemented on [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331), as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15), which uses the [**Transmission Control Protocol (TCP)**](#gt_b08d36f6-b5c6-4ce4-8d2d-6f2ab75ea4cb) as a transport. Aside from managing the policy for the firewall itself, this protocol is used to remotely manage the security policy database of the Security Architecture for the Internet Protocol [[RFC4301]](https://go.microsoft.com/fwlink/?LinkId=90465), which describes how [**Internet Protocol Security (IPsec)**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) should be enforced and what options the [**Internet Key Exchange (IKE)**](#gt_294fef97-5790-4d41-971e-dd255b783e68) [[RFC2409]](https://go.microsoft.com/fwlink/?LinkId=90349), [**Authenticated IP (AuthIP)**](#gt_3791f3e1-cf2f-4605-9fcc-54f526f036cf) [[MS-AIPS]](%5bMS-AIPS%5d.pdf#Section_eee3de6438474451978e9513ff187d30), and [**Internet Key Exchange (IKEv2)**](#gt_2cd83a7f-fc17-4283-b3f3-59feb25114bf) [[RFC4306]](https://go.microsoft.com/fwlink/?LinkId=90469) have available to negotiate. This protocol also exposes an abstract interface to configure firewall and advanced security policy for use by other mechanisms such as [**Group Policy**](#gt_defe8c22-1365-4e5e-abf7-46ad112d3bda) [[MS-GPFAS]](%5bMS-GPFAS%5d.pdf#Section_46e8d583a4ce4c43b399566afb1eec7f).

## Prerequisites/Preconditions

This protocol assumes that the firewall and advanced security components have been initialized, are running, and have registered the corresponding [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) interface that is defined in section [2.1](#Section_81eb95d6df6349b6905265c99664e71f). This protocol also assumes that the policy in the host firewall and advanced security components, which resides on the server side, already allows the inbound traffic that the client computer, which is running the management tool, sends to the server during exercise of this protocol.

This protocol requires [**Security Support Provider Interface (SSPI)**](#gt_fb216516-748b-4873-8bdd-64c5f4da9920) security by using packet privacy protection level (RPC\_C\_PROTECT\_LEVEL\_PKT\_PRIVACY) and GSS negotiate authentication (RPC\_C\_AUTHN\_GSS\_NEGOTIATE), which negotiates between Kerberos Protocol Extensions [[MS-KILE]](%5bMS-KILE%5d.pdf#Section_2a32282edd484ad9a542609804b02cc9) and NT LAN Manager (NTLM) Authentication Protocol [[MS-NLMP]](%5bMS-NLMP%5d.pdf#Section_b38c36ed28044868a9ff8dd3182128e4) authentication.

## Applicability Statement

This protocol is used to address the needs defined in section [1.3](#Section_3e3c5b45430c43c684421bba3853f9d9).

## Versioning and Capability Negotiation

This document covers versioning and capability negotiation issues in the following areas:

* Supported Transports: This protocol uses a single [**RPC protocol sequence**](#gt_0c171cc7-e9c4-41b6-95a9-536db0042c7a), as specified in section [2.1](#Section_81eb95d6df6349b6905265c99664e71f).
* Protocol Versions: This protocol has only one interface version. There are also several policy versions, which can be tied to policies and specific policy objects, as defined in section [2.2](#Section_2f45305aaf9249678d918f137ebc11e0). The policy versions are 0x0200, 0x0201, 0x020A, 0x0214, 0x0216, 0x0218, 0x0219, 0x021A, and 0x021B.[<1>](#Appendix_A_1" \o "Product behavior note 1) Protocol Versions are used as Binary Versions and Schema Versions (also called policy versions).

The policy versions listed above can be translated into binary versions by considering the two-byte values to consist of a "high byte" and a "low byte". Convert each byte to decimal and separate them with a period (".") to obtain the binary version. For example, the policy version 0x0214 is mapped to binary version 2.20. Schema versions are similar to binary versions but with an underscore ("\_") instead of a period.

* Security and Authentication Methods: This protocol supports both Kerberos Protocol Extensions [[MS-KILE]](%5bMS-KILE%5d.pdf#Section_2a32282edd484ad9a542609804b02cc9) and NT LAN Manager (NTLM) Authentication Protocol [[MS-NLMP]](%5bMS-NLMP%5d.pdf#Section_b38c36ed28044868a9ff8dd3182128e4) authentication methods, section 2.1.
* Localization: This protocol passes text strings without considering localization. However, some strings can be formatted in such a way that the firewall component knows where to look for localized versions of these strings, as defined in section 2.2. These strings can also be resolved with specific flags and method calls, as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7).
* Capability Negotiation: A configuration option defined in section [2.2.41](#Section_faf4ffbe1d5140adae902230f2c0b6a9) contains the maximum policy version and the binary supported by the server. With this option, a client can understand what can and cannot be expressed in this protocol and the methods that are supported to do so. The data types in section 2.2 and the existence and behavior of methods in section 3.1.4 are defined in terms of these policy versions when appropriate. No other negotiation capabilities, version-specific or otherwise, are present in this protocol.
* Byte order: All values defined in this specification are independent of whether the platform uses big-endian or little-endian byte order. For instance, protocol version 0x0200 = 512 decimal, and will be value 512 (0x0200) on both little-endian and big-endian platforms. Marshaling any values defined within this specification is handled by [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) (see [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15)).

## Vendor-Extensible Fields

This protocol uses Win32 error codes. These values are taken from the Windows error number space that is specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). Vendors SHOULD reuse those values with their indicated meaning. Choosing any other value runs the risk of a collision in the future.

This protocol uses NTSTATUS values, as specified in [MS-ERREF]. Vendors can choose their own values for this field provided that the C bit (0x20000000) is set, indicating that it is a customer code.

Currently, vendors are not expected to extend this protocol. Therefore, the protocol does not consider provisions for extensions by parties other than Microsoft.

## Standards Assignments

| Parameter | Value | Reference |
| --- | --- | --- |
| [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) interface [**UUID**](#gt_c4813fc3-b2e5-4aa3-bde7-421d950d68d3) for the Firewall and Advanced Security Protocol | 6b5bdd1e-528c-422c-af8c-a4079be4fe48 | Section [2.1](#Section_81eb95d6df6349b6905265c99664e71f) |

No standards assignments have been received for this protocol. All values used in these extensions are in private ranges specified in section 2.1. This protocol uses RPC [**dynamic endpoints**](#gt_46da887f-3f66-4941-a854-e51c52cf4c56), as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) chapters 6, 7, 8, 9, 10, 11, 12, 13, and 14.

# Messages

## Transport

This protocol uses the [**Remote Procedure Call (RPC)**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) over [**TCP**](#gt_b08d36f6-b5c6-4ce4-8d2d-6f2ab75ea4cb). It also uses RPC [**dynamic endpoints**](#gt_46da887f-3f66-4941-a854-e51c52cf4c56), as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) chapters 6, 7, 8, 9, 10, 11, 12, 13, and 14.

This RPC protocol MUST use [**Security Support Provider Interface (SSPI)**](#gt_fb216516-748b-4873-8bdd-64c5f4da9920) security by using packet privacy protection level (RPC\_C\_PROTECT\_LEVEL\_PKT\_PRIVACY) and GSS negotiate authentication (RPC\_C\_AUTHN\_GSS\_NEGOTIATE), which negotiates between Kerberos Protocol Extensions, as specified in [[MS-KILE]](%5bMS-KILE%5d.pdf#Section_2a32282edd484ad9a542609804b02cc9), and NT LAN Manager (NTLM) Authentication Protocol, as specified in [[MS-NLMP]](%5bMS-NLMP%5d.pdf#Section_b38c36ed28044868a9ff8dd3182128e4) authentication.

This protocol MUST use the following interface identifier as specified in [C706] section 3.1.9:

| uuid: | 6b5bdd1e-528c-422c-af8c-a4079be4fe48 |
| --- | --- |
| vers\_major: | 1 |
| vers\_minor: | 0 |

The server MUST register this interface identifier with the RPC run-time during server initialization as specified in section [3.1.3](#Section_e8924ac5aa4a41d1bf654f46b3d399aa). The client MUST use this interface identifier when binding to the RPC server as specified in section [3.2.3](#Section_27cba968dfb34c91b36b1e615550eaa1).

## Common Data Types

In addition to [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) base types and definitions specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) and [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2), additional data types are defined in the sections that follow.

### FW\_STORE\_TYPE

This data type defines enumerations used to identify store types.

1. typedef enum \_tag\_FW\_STORE\_TYPE
2. {
3. FW\_STORE\_TYPE\_INVALID,
4. FW\_STORE\_TYPE\_GP\_RSOP,
5. FW\_STORE\_TYPE\_LOCAL,
6. FW\_STORE\_TYPE\_NOT\_USED\_VALUE\_3,
7. FW\_STORE\_TYPE\_NOT\_USED\_VALUE\_4,
8. FW\_STORE\_TYPE\_DYNAMIC,
9. FW\_STORE\_TYPE\_GPO,
10. FW\_STORE\_TYPE\_DEFAULTS,
11. FW\_STORE\_TYPE\_NOT\_USED\_VALUE\_8,
12. FW\_STORE\_TYPE\_NOT\_USED\_VALUE\_9,
13. FW\_STORE\_TYPE\_NOT\_USED\_VALUE\_10,
14. FW\_STORE\_TYPE\_NOT\_USED\_VALUE\_11,
15. FW\_STORE\_TYPE\_MAX
16. } FW\_STORE\_TYPE;

**FW\_STORE\_TYPE\_INVALID:** This value is invalid and MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of zero.

**FW\_STORE\_TYPE\_GP\_RSOP:** This value identifies the store that contains all the policies from the different [**Group Policy Objects (GPOs)**](#gt_dec32233-8776-4151-91a0-8624a2b9abb0) that contain the networkwide policy. This store is persisted in the registry. It is downloaded by the [**Group Policy**](#gt_defe8c22-1365-4e5e-abf7-46ad112d3bda) component (for more information, see [[MS-GPREG]](%5bMS-GPREG%5d.pdf#Section_834da877264f45899b80b6b012c8edc3)) and read by the firewall and advanced security components; therefore, it is a read-only store. This symbolic constant has a value of 1.

**FW\_STORE\_TYPE\_LOCAL:** This value identifies the store that contains the local host policy. This store is persisted in the registry by the firewall and advanced security components; therefore, it is a read/write store. This symbolic constant has a value of 2.

**FW\_STORE\_TYPE\_NOT\_USED\_VALUE\_3:** This store is currently not used over the wire. This symbolic constant has a value of 3.

**FW\_STORE\_TYPE\_NOT\_USED\_VALUE\_4:** This store is currently not used over the wire. This symbolic constant has a value of 4.

**FW\_STORE\_TYPE\_DYNAMIC:** This value identifies the store that contains the effective policy, that is, the aggregated and merged policy from all policy sources. Policy objects can be added and modified on this store, but they are not persisted and will be lost the next time the firewall and advanced security components initialize. Policy objects on this store can be modified only if they were originally added to this store. This symbolic constant has a value of 5.

**FW\_STORE\_TYPE\_GPO:** This value is not used on the wire. This symbolic constant has a value of 6.

**FW\_STORE\_TYPE\_DEFAULTS:** This value identifies the store that contains the defaults that the host operating system had out-of-box. This store is persisted in the registry. It is written by the host operating system setup. It is read by the firewall and advanced security components when it is instructed to go back to the default out-of-box configuration; hence it is a read-only store. This symbolic constant has a value of 7.

**FW\_STORE\_TYPE\_NOT\_USED\_VALUE\_8:** This store is currently not used over the wire. This symbolic constant has a value of 8.

**FW\_STORE\_TYPE\_NOT\_USED\_VALUE\_9:** This store is currently not used over the wire. This symbolic constant has a value of 9.

**FW\_STORE\_TYPE\_NOT\_USED\_VALUE\_10:** This store is currently not used over the wire. This symbolic constant has a value of 10.

**FW\_STORE\_TYPE\_NOT\_USED\_VALUE\_11:** This store is currently not used over the wire. This symbolic constant has a value of 11.

**FW\_STORE\_TYPE\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. This symbolic constant is defined for simplicity in writing IDL definitions and code. It has a value of 8.

### FW\_PROFILE\_TYPE

This data type defines the enumerations that are used to identify profile types. The enumeration values are bitmasks. Implementations MUST support using a single bitmask value and MUST support a combination of bitmask values. Valid combinations of bitmask values are all possible combinations using FW\_PROFILE\_TYPE\_DOMAIN, FW\_PROFILE\_TYPE\_PRIVATE, FW\_PROFILE\_TYPE\_PUBLIC, and FW\_PROFILE\_TYPE\_ALL. A profile is a set of networks to which a firewall policy might apply.

1. typedef [v1\_enum] enum \_tag\_FW\_PROFILE\_TYPE
2. {
3. FW\_PROFILE\_TYPE\_INVALID = 0x000,
4. FW\_PROFILE\_TYPE\_DOMAIN = 0x001,
5. FW\_PROFILE\_TYPE\_STANDARD = 0x002,
6. FW\_PROFILE\_TYPE\_PRIVATE = 0x002,
7. FW\_PROFILE\_TYPE\_PUBLIC = 0x004,
8. FW\_PROFILE\_TYPE\_ALL = 0x7FFFFFFF,
9. FW\_PROFILE\_TYPE\_CURRENT = 0x80000000,
10. FW\_PROFILE\_TYPE\_NONE = 0x80000001
11. } FW\_PROFILE\_TYPE;

**FW\_PROFILE\_TYPE\_INVALID:** This value is invalid and MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code.

**FW\_PROFILE\_TYPE\_DOMAIN:** This value represents the profile for networks that are connected to domains.

**FW\_PROFILE\_TYPE\_STANDARD:** This value represents the standard profile for networks. These networks are classified as private by the administrators in the server host. The classification happens the first time the host connects to the network. Usually these networks are behind Network Address Translation (NAT) devices, routers, and other edge devices, and they are in a private location, such as a home or an office.

**FW\_PROFILE\_TYPE\_PRIVATE:** This value represents the profile for private networks, which is represented by the same value as that used for FW\_PROFILE\_TYPE\_STANDARD.

**FW\_PROFILE\_TYPE\_PUBLIC:** This value represents the profile for public networks. These networks are classified as public by the administrators in the server host. The classification happens the first time the host connects to the network. Usually these networks are those at airports, coffee shops, and other public places where the peers in the network or the network administrator are not trusted.

**FW\_PROFILE\_TYPE\_ALL:** This value represents all these network sets and any future network sets.

**FW\_PROFILE\_TYPE\_CURRENT:** This value represents the current profiles to which the firewall and advanced security components determine the host is connected at the moment of the call. This value can be specified only in method calls, and it cannot be combined with other flags.

**FW\_PROFILE\_TYPE\_NONE:** This value represents no profile and is invalid. It is defined for simplicity in writing IDL definitions and code. This and greater values MUST NOT be used.

### FW\_POLICY\_ACCESS\_RIGHT

This enumeration defines access rights for the policy elements that can be accessed using the Firewall and Advanced Security Protocol. The values are not bitmasks and SHOULD NOT be used in bitwise OR operations.

1. typedef enum \_tag\_FW\_POLICY\_ACCESS\_RIGHT
2. {
3. FW\_POLICY\_ACCESS\_RIGHT\_INVALID,
4. FW\_POLICY\_ACCESS\_RIGHT\_READ,
5. FW\_POLICY\_ACCESS\_RIGHT\_READ\_WRITE,
6. FW\_POLICY\_ACCESS\_RIGHT\_MAX
7. } FW\_POLICY\_ACCESS\_RIGHT;

**FW\_POLICY\_ACCESS\_RIGHT\_INVALID:** This value is invalid and MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of zero.

**FW\_POLICY\_ACCESS\_RIGHT\_READ:** This value represents a read-only access right. This symbolic constant has a value of 1.

**FW\_POLICY\_ACCESS\_RIGHT\_READ\_WRITE:** This value represents a read and write access right. This symbolic constant has a value of 2.

**FW\_POLICY\_ACCESS\_RIGHT\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. This symbolic constant is defined for simplicity in writing IDL definitions and code. It has a value of 3.

### FW\_IPV4\_SUBNET

This structure defines IPv4 subnets. It is used in policy rules.

1. typedef struct \_tag\_FW\_IPV4\_SUBNET {
2. unsigned long dwAddress;
3. unsigned long dwSubNetMask;
4. } FW\_IPV4\_SUBNET,
5. \*PFW\_IPV4\_SUBNET;

**dwAddress:**   This field represents the IPv4 address.

**dwSubNetMask:**  This field contains the subnet mask in host network order. If it contains ones, they MUST be contiguous and shifted to the most significant bits.

A **dwSubNetMask** of 0x00000000 is invalid. A subnet mask of 0xFFFFFFFF means that the subnet mask represents a single address.

### FW\_IPV4\_SUBNET\_LIST

This structure is used to contain a number of [FW\_IPV4\_SUBNET](#Section_ed0a65ecd3b6461ca4caecf08ed18025) elements.

1. typedef struct \_tag\_FW\_IPV4\_SUBNET\_LIST {
2. [range(0, 1000)] unsigned long dwNumEntries;
3. [size\_is(dwNumEntries)] PFW\_IPV4\_SUBNET pSubNets;
4. } FW\_IPV4\_SUBNET\_LIST,
5. \*PFW\_IPV4\_SUBNET\_LIST;

**dwNumEntries:**  This field specifies the number of subnets that the structure contains.

**pSubNets:**   A pointer to an array of FW\_IPV4\_SUBNET elements. The number of elements is given by **dwNumEntries**.

### FW\_IPV6\_SUBNET

This structure represents an IPv6 subnet.

1. typedef struct \_tag\_FW\_IPV6\_SUBNET {
2. unsigned char Address[16];
3. [range(0, 128)] unsigned long dwNumPrefixBits;
4. } FW\_IPV6\_SUBNET,
5. \*PFW\_IPV6\_SUBNET;

**Address:**  This field contains a 16-octet IPv6 address.

**dwNumPrefixBits:**  This field contains the number of more-significant bits that represent the IPv6 subnet.

The **dwNumPrefixBits** MUST NOT be greater than 128 and not less than 1. The address SHOULD NOT be an unspecified address (an address composed of all zeros),[<2>](#Appendix_A_2" \o "Product behavior note 2) and it MUST not be a loopback address.

### FW\_IPV6\_SUBNET\_LIST

This structure is used to contain a number of [FW\_IPV6\_SUBNET](#Section_ccb501eb66204b49befd2b59828a28c8) elements.

1. typedef struct \_tag\_FW\_IPV6\_SUBNET\_LIST {
2. [range(0, 1000)] unsigned long dwNumEntries;
3. [size\_is(dwNumEntries)] PFW\_IPV6\_SUBNET pSubNets;
4. } FW\_IPV6\_SUBNET\_LIST,
5. \*PFW\_IPV6\_SUBNET\_LIST;

**dwNumEntries:**  This field specifies the number of subnets that the structure contains.

**pSubNets:**  A pointer to an array of FW\_IPV6\_SUBNET elements. The number of elements is given by **dwNumEntries**.

### FW\_IPV4\_ADDRESS\_RANGE

This structure represents a range of IPv4 addresses within the IPv4 address space.

1. typedef struct \_tag\_FW\_IPV4\_ADDRESS\_RANGE {
2. unsigned long dwBegin;
3. unsigned long dwEnd;
4. } FW\_IPV4\_ADDRESS\_RANGE,
5. \*PFW\_IPV4\_ADDRESS\_RANGE;

**dwBegin:**  The first IPv4 address of the range in the IPv4 address space defined by this structure. The address is included in the range.

**dwEnd:**  The last IPv4 address of the range in the IPv4 address space defined by this structure. The address is included in the range.

Valid FW\_IPV4\_ADDRESS\_RANGE structures MUST have a **dwBegin** value less than or equal to the **dwEnd** value. Structures with **dwBegin** equal to **dwEnd** represent a single IPv4 address.

### FW\_IPV4\_RANGE\_LIST

This structure is used to contain a number of [FW\_IPV4\_ADDRESS\_RANGE](#Section_f11b849900a043d8bf3a6b596ec0d55c) elements.

1. typedef struct \_tag\_FW\_IPV4\_RANGE\_LIST {
2. [range(0, 1000)] unsigned long dwNumEntries;
3. [size\_is(dwNumEntries)] PFW\_IPV4\_ADDRESS\_RANGE pRanges;
4. } FW\_IPV4\_RANGE\_LIST,
5. \*PFW\_IPV4\_RANGE\_LIST;

**dwNumEntries:**  This field specifies the number of IPv4 address ranges that the structure contains.

**pRanges:**  A pointer to an array of FW\_IPV4\_ADDRESS\_RANGE elements. The number of elements is given by **dwNumEntries**.

### FW\_IPV6\_ADDRESS\_RANGE

This structure represents a range of IPv6 addresses within the IPv6 address space.

1. typedef struct \_tag\_FW\_IPV6\_ADDRESS\_RANGE {
2. unsigned char Begin[16];
3. unsigned char End[16];
4. } FW\_IPV6\_ADDRESS\_RANGE,
5. \*PFW\_IPV6\_ADDRESS\_RANGE;

**Begin:**  A 16-octet array containing the first IPv6 address of the range in the IPv6 address range defined by this structure.

**End:**  A 16-octet array containing the last IPv6 address of the range in the IPv6 address range defined by this structure.

Valid FW\_IPV6\_ADDRESS\_RANGE structures MUST have a **Begin** value less than or equal to the **End** value. Structures with **Begin** equal to **End** represent a single IPv6 address. **Begin** and **End** MUST NOT contain either an unspecified or a loopback address.

**Begin** and **End** are in network order.

### FW\_IPV6\_RANGE\_LIST

This structure is used to contain a number of [FW\_IPV6\_ADDRESS\_RANGE](#Section_fbf349cc8455490db97d951c63084861) elements.

1. typedef struct \_tag\_FW\_IPV6\_RANGE\_LIST {
2. [range(0, 1000)] unsigned long dwNumEntries;
3. [size\_is(dwNumEntries)] PFW\_IPV6\_ADDRESS\_RANGE pRanges;
4. } FW\_IPV6\_RANGE\_LIST,
5. \*PFW\_IPV6\_RANGE\_LIST;

**dwNumEntries:**  This field specifies the number of IPv6 address ranges that the structure contains.

**pRanges:**  A pointer to an array of FW\_IPV6\_ADDRESS\_RANGE elements. The number of elements is given by **dwNumEntries**.

### FW\_PORT\_RANGE

This structure represents a range of ports. Ports are 16-bit unsigned values used in [**TCP**](#gt_b08d36f6-b5c6-4ce4-8d2d-6f2ab75ea4cb) and UDP protocols.

1. typedef struct \_tag\_FW\_PORT\_RANGE {
2. unsigned short wBegin;
3. unsigned short wEnd;
4. } FW\_PORT\_RANGE,
5. \*PFW\_PORT\_RANGE;

**wBegin:**  This field specifies the first port included in the range defined.

**wEnd:**  This field specifies the last port included in the range defined.

Valid FW\_PORT\_RANGE structures MUST have a **wBegin** value less than or equal to the **wEnd** value. In this protocol, **wBegin** is equal to **wEnd**.

### FW\_PORT\_RANGE\_LIST

This structure is used to contain a number of [FW\_PORT\_RANGE](#Section_3122d4b1950043af900a27fe64688623) elements.

1. typedef struct \_tag\_FW\_PORT\_RANGE\_LIST {
2. [range(0, 1000)] unsigned long dwNumEntries;
3. [size\_is(dwNumEntries)] PFW\_PORT\_RANGE pPorts;
4. } FW\_PORT\_RANGE\_LIST,
5. \*PFW\_PORT\_RANGE\_LIST;

**dwNumEntries:**  This field specifies the number of port ranges that the structure contains.

**pPorts:**  A pointer to an array of FW\_PORT\_RANGE elements. The number of elements is given as **dwNumEntries**.

### FW\_PORT\_KEYWORD

This enumeration identifies (with bitmask flags) the ports used by specific well-known protocols. The ports corresponding to these keywords change dynamically and are tracked by the **PortsInUse** object (see section [3.1.1](#Section_43507d538955416db913dfb27dc76b17)). All the flags supported by a given schema version can be combined, except for the restrictions placed on the **wPortKeywords** field as stated in [FW\_RULE (section 2.2.36)](#Section_8c008258166d46d49090f2ffaa01be4b) and [FW\_CS\_RULE (section 2.2.54)](#Section_0d0641105f2e4b68aa63032c6cd5e4c6).

1. typedef enum \_tag\_FW\_PORT\_KEYWORD
2. {
3. FW\_PORT\_KEYWORD\_NONE = 0x00,
4. FW\_PORT\_KEYWORD\_DYNAMIC\_RPC\_PORTS = 0x01,
5. FW\_PORT\_KEYWORD\_RPC\_EP = 0x02,
6. FW\_PORT\_KEYWORD\_TEREDO\_PORT = 0x04,
7. FW\_PORT\_KEYWORD\_IP\_TLS\_IN = 0x08,
8. FW\_PORT\_KEYWORD\_IP\_TLS\_OUT = 0x10,
9. FW\_PORT\_KEYWORD\_DHCP = 0x20,
10. FW\_PORT\_KEYWORD\_PLAYTO\_DISCOVERY = 0x40,
11. FW\_PORT\_KEYWORD\_MAX = 0x80,
12. FW\_PORT\_KEYWORD\_MAX\_V2\_1 = 0x08,
13. FW\_PORT\_KEYWORD\_MAX\_V2\_10 = 0x20
14. } FW\_PORT\_KEYWORD;

**FW\_PORT\_KEYWORD\_NONE:** Specifies that no port keywords are used.

**FW\_PORT\_KEYWORD\_DYNAMIC\_RPC\_PORTS:** Represents all ports in the **PortsInUse** collection where **IsDynamicRPC** is true.

**FW\_PORT\_KEYWORD\_RPC\_EP:** Represents all ports in the **PortsInUse** collection where **IsRPCEndpointMapper** is true.

**FW\_PORT\_KEYWORD\_TEREDO\_PORT:** Represents all ports in the **PortsInUse** collection where **IsTeredo** is true.

**FW\_PORT\_KEYWORD\_IP\_TLS\_IN:** Represents all ports in the **PortsInUse** collection where **IsIPTLSIn** is true. For schema versions 0x0200 and 0x0201, this value is invalid and MUST NOT be used. This symbolic constant has a value of 0x08.

**FW\_PORT\_KEYWORD\_IP\_TLS\_OUT:** Represents all ports in the **PortsInUse** collection where **IsIPTLSOut** is true. For schema versions 0x0200 and 0x0201, this value is invalid and MUST NOT be used. This symbolic constant has a value of 0x10.

**FW\_PORT\_KEYWORD\_DHCP:** Represents all ports in the PortsInUse collection where IsDHCPClient is true. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used. This symbolic constant has a value of 0x20.

**FW\_PORT\_KEYWORD\_PLAYTO\_DISCOVERY:** Represents all ports in the PortsInUse collection where IsPlayToDiscovery is true. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used. This symbolic constant has a value of 0x40.

**FW\_PORT\_KEYWORD\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 0x80.

**FW\_PORT\_KEYWORD\_MAX\_V2\_1:** This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x0201 and earlier. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 0x08.

**FW\_PORT\_KEYWORD\_MAX\_V2\_10:** This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x020A and earlier. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 0x20.

### FW\_PORTS

This structure contains the ports represented statically through [FW\_PORT\_RANGE](#Section_3122d4b1950043af900a27fe64688623) structures or symbolically through [FW\_PORT\_KEYWORD](#Section_f2a500d8b8cb48e1b4f21f85c89011ad) enumeration values.

1. typedef struct \_tag\_FW\_PORTS {
2. unsigned short wPortKeywords;
3. FW\_PORT\_RANGE\_LIST Ports;
4. } FW\_PORTS,
5. \*PFW\_PORTS;

**wPortKeywords:**  This field is a combination of FW\_PORT\_KEYWORDS.

**Ports:**  This field is a list of specifically defined ports.

### FW\_ICMP\_TYPE\_CODE

This data type defines ICMP (internet control message protocol with protocol numbers assigned in [[IANA-PROTO-NUM]](https://go.microsoft.com/fwlink/?LinkId=89889)) message types and codes. It specifies an ICMP type and either its specific code or all codes for that type.

1. typedef struct \_tag\_FW\_ICMP\_TYPE\_CODE {
2. unsigned char bType;
3. [range(0, 256)] unsigned short wCode;
4. } FW\_ICMP\_TYPE\_CODE,
5. \*PFW\_ICMP\_TYPE\_CODE;

**bType:**  This field specifies the ICMP type.

**wCode:**  This field specifies the ICMP code.

The **wCode** field MUST contain values between 0x0000 and 0x0100. When **wCode** contains 0x100, it expresses any ICMP code belonging to the corresponding ICMP type. When **wCode** contains values in the range 0 to 0x00FF, it expresses a specific ICMP code.

All valid ICMP type and code combinations are valid, even those not currently assigned for a specific use.

### FW\_ICMP\_TYPE\_CODE\_LIST

This structure is used to contain a number of [FW\_ICMP\_TYPE\_CODE](#Section_6feb51f6495e4396b86c0dbfc5650dff) elements.

1. typedef struct \_tag\_FW\_ICMP\_TYPE\_CODE\_LIST {
2. [range(0, 1000)] unsigned long dwNumEntries;
3. [size\_is(dwNumEntries)] PFW\_ICMP\_TYPE\_CODE pEntries;
4. } FW\_ICMP\_TYPE\_CODE\_LIST,
5. \*PFW\_ICMP\_TYPE\_CODE\_LIST;

**dwNumEntries:**  This field specifies the number of FW\_ICMP\_TYPE\_CODE elements that the structure contains.

**pEntries:**  A pointer to an array of FW\_ICMP\_TYPE\_CODE elements. The number of elements is given by **dwNumEntries**.

### FW\_INTERFACE\_LUIDS

This structure is used to contain [**locally unique identifier (LUID)**](#gt_96b64af9-1896-4bde-b988-54d469c5affd) values that uniquely represent single network adapters (NICs) within a specific computer.

1. typedef struct \_tag\_FW\_INTERFACE\_LUIDS {
2. [range(0, 1000)] unsigned long dwNumLUIDs;
3. [size\_is(dwNumLUIDs)] GUID\* pLUIDs;
4. } FW\_INTERFACE\_LUIDS,
5. \*PFW\_INTERFACE\_LUIDS;

**dwNumLUIDs:**  This field specifies the number of interface LUIDs that the structure contains.

**pLUIDs:**  A pointer to an array of [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) elements. The number of elements is given by **dwNumLUIDs**. The GUID data type is specified in [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2).

### FW\_DIRECTION

This enumeration represents the direction of network traffic flow.

1. typedef enum \_tag\_FW\_DIRECTION
2. {
3. FW\_DIR\_INVALID = 0,
4. FW\_DIR\_IN,
5. FW\_DIR\_OUT,
6. FW\_DIR\_MAX
7. } FW\_DIRECTION;

**FW\_DIR\_INVALID:** This is an invalid value, and it MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of zero.

**FW\_DIR\_IN:** Specifies an inbound network traffic flow. These are flows that are initiated by a remote machine toward the local machine. This symbolic constant has a value of 1.

**FW\_DIR\_OUT:** Specifies an outbound network traffic flow. These are flows that are initiated by the local machine toward a remote machine. This symbolic constant has a value of 2.

**FW\_DIR\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. This symbolic constant is defined for simplicity in writing IDL definitions and code. It has a value of 3.

### FW\_INTERFACE\_TYPE

This enumeration is used to represent types of network adapters (NICs) in a specific machine. Each type might have one or more network adapters.

1. typedef enum \_tag\_FW\_INTERFACE\_TYPE
2. {
3. FW\_INTERFACE\_TYPE\_ALL = 0x0000,
4. FW\_INTERFACE\_TYPE\_LAN = 0x0001,
5. FW\_INTERFACE\_TYPE\_WIRELESS = 0x0002,
6. FW\_INTERFACE\_TYPE\_REMOTE\_ACCESS = 0x0004,
7. FW\_INTERFACE\_TYPE\_MAX = 0x0008
8. } FW\_INTERFACE\_TYPE;

**FW\_INTERFACE\_TYPE\_ALL:** Represents all types of network adapters (NICs). The following types fall into this type.

**FW\_INTERFACE\_TYPE\_LAN:** Represents network adapters (NICs) that use wired network physical layers such as Ethernet.

**FW\_INTERFACE\_TYPE\_WIRELESS:** Represents network adapters that use the wireless 802 network physical layer.

**FW\_INTERFACE\_TYPE\_REMOTE\_ACCESS:** Represents network adapters that use VPN connections.

**FW\_INTERFACE\_TYPE\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 0x0008.

### FW\_ADDRESS\_KEYWORD

This enumeration is used to represent specific address types. As specified in the following descriptions, these address types can change dynamically.

1. typedef enum \_tag\_FW\_ADDRESS\_KEYWORD
2. {
3. FW\_ADDRESS\_KEYWORD\_NONE = 0x0000,
4. FW\_ADDRESS\_KEYWORD\_LOCAL\_SUBNET = 0x0001,
5. FW\_ADDRESS\_KEYWORD\_DNS = 0x0002,
6. FW\_ADDRESS\_KEYWORD\_DHCP = 0x0004,
7. FW\_ADDRESS\_KEYWORD\_WINS = 0x0008,
8. FW\_ADDRESS\_KEYWORD\_DEFAULT\_GATEWAY = 0x0010,
9. FW\_ADDRESS\_KEYWORD\_INTRANET = 0x0020,
10. FW\_ADDRESS\_KEYWORD\_INTERNET = 0x0040,
11. FW\_ADDRESS\_KEYWORD\_PLAYTO\_RENDERERS = 0x0080,
12. FW\_ADDRESS\_KEYWORD\_REMOTE\_INTRANET = 0x0100,
13. FW\_ADDRESS\_KEYWORD\_MAX = 0x0200,
14. FW\_ADDRESS\_KEYWORD\_MAX\_V2\_10 = 0x0020
15. } FW\_ADDRESS\_KEYWORD;

**FW\_ADDRESS\_KEYWORD\_NONE:** Specifies that no specific keyword is used.

**FW\_ADDRESS\_KEYWORD\_LOCAL\_SUBNET:** Represents the collection of addresses that are currently within the local subnet of the computer.

**FW\_ADDRESS\_KEYWORD\_DNS:** Represents the collection of addresses of the current DNS servers.

**FW\_ADDRESS\_KEYWORD\_DHCP:** Represents the collection of addresses of the current DHCP servers.

**FW\_ADDRESS\_KEYWORD\_WINS:** Represents the collection of addresses of the current WINS servers.

**FW\_ADDRESS\_KEYWORD\_DEFAULT\_GATEWAY:** Represents the collection of addresses of the current gateway servers.

**FW\_ADDRESS\_KEYWORD\_INTRANET:** Represents the collection of addresses that are currently within the local intranet of the computer. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used.

**FW\_ADDRESS\_KEYWORD\_INTERNET:** Represents the collection of addresses that are currently not within the local intranet or remote intranet of the computer. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used.

**FW\_ADDRESS\_KEYWORD\_PLAYTO\_RENDERERS:** Represents the collection of addresses of the current Digital Media Renderer devices as defined in [[MS-DLNHND]](%5bMS-DLNHND%5d.pdf#Section_a12b022ab79346b2aaa96100611b5038) section 3.3. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used.

**FW\_ADDRESS\_KEYWORD\_REMOTE\_INTRANET:** Represents the collection of addresses that are currently within the remote intranet of the computer. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used.

**FW\_ADDRESS\_KEYWORD\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 0x0200.

**FW\_ADDRESS\_KEYWORD\_MAX\_V2\_10:** This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x020A and earlier. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 0x0020.

### FW\_ADDRESSES

This structure contains a list of address structures. Static and symbolic representations are supported, but a structure can contain only one representation type. The address structure representations follow:

Static Representation

* [FW\_IPV4\_SUBNET\_LIST](#Section_a230f0e48005498389e3e417b3354ae5)
* [FW\_IPV4\_RANGE\_LIST](#Section_de836948bf68424d906d406723bd0deb)
* [FW\_IPV6\_SUBNET\_LIST](#Section_a8d05086bc48453eaebfaa11fb95b9bd)
* [FW\_IPV6\_RANGE\_LIST](#Section_062373f2c3dc49a98b3f960d2a88332d)

Symbolic Representation

* [FW\_ADDRESS\_KEYWORD](#Section_d69ec3fe85074524bdcc813cbb3bf85f) enumeration values

The FW\_ADDRESSES definition follows:

1. typedef struct \_tag\_FW\_ADDRESSES {
2. unsigned long dwV4AddressKeywords;
3. unsigned long dwV6AddressKeywords;
4. FW\_IPV4\_SUBNET\_LIST V4SubNets;
5. FW\_IPV4\_RANGE\_LIST V4Ranges;
6. FW\_IPV6\_SUBNET\_LIST V6SubNets;
7. FW\_IPV6\_RANGE\_LIST V6Ranges;
8. } FW\_ADDRESSES,
9. \*PFW\_ADDRESSES;

**dwV4AddressKeywords:**  A combination of FW\_ADDRESS\_KEYWORD flags. Addresses in this field are specified from the IPv4 address space.

**dwV6AddressKeywords:**  A combination of FW\_ADDRESS\_KEYWORD flags. Addresses in this field are specified from the IPv6 address space.

**V4SubNets:**  A list of specifically defined IPv4 address subnets.

**V4Ranges:**  A list of specifically defined IPv4 address ranges.

**V6SubNets:**  A list of specifically defined IPv6 address subnets.

**V6Ranges:**  A list of specifically defined IPv6 address ranges.

### FW\_RULE\_STATUS

This enumeration represents status codes that identify the error states of a policy object, including successful states. If an object is in an erroneous state, the enumeration value represents a reason for the error.

1. typedef [v1\_enum] enum \_tag\_FW\_RULE\_STATUS
2. {
3. FW\_RULE\_STATUS\_OK = 0x00010000,
4. FW\_RULE\_STATUS\_PARTIALLY\_IGNORED = 0x00020000,
5. FW\_RULE\_STATUS\_IGNORED = 0x00040000,
6. FW\_RULE\_STATUS\_PARSING\_ERROR = 0x00080000,
7. FW\_RULE\_STATUS\_PARSING\_ERROR\_NAME = 0x00080001,
8. FW\_RULE\_STATUS\_PARSING\_ERROR\_DESC = 0x00080002,
9. FW\_RULE\_STATUS\_PARSING\_ERROR\_APP = 0x00080003,
10. FW\_RULE\_STATUS\_PARSING\_ERROR\_SVC = 0x00080004,
11. FW\_RULE\_STATUS\_PARSING\_ERROR\_RMA = 0x00080005,
12. FW\_RULE\_STATUS\_PARSING\_ERROR\_RUA = 0x00080006,
13. FW\_RULE\_STATUS\_PARSING\_ERROR\_EMBD = 0x00080007,
14. FW\_RULE\_STATUS\_PARSING\_ERROR\_RULE\_ID = 0x00080008,
15. FW\_RULE\_STATUS\_PARSING\_ERROR\_PHASE1\_AUTH = 0x00080009,
16. FW\_RULE\_STATUS\_PARSING\_ERROR\_PHASE2\_CRYPTO = 0x0008000A,
17. FW\_RULE\_STATUS\_PARSING\_ERROR\_PHASE2\_AUTH = 0x0008000B,
18. FW\_RULE\_STATUS\_PARSING\_ERROR\_RESOLVE\_APP = 0x0008000C,
19. FW\_RULE\_STATUS\_PARSING\_ERROR\_MAINMODE\_ID = 0x0008000D,
20. FW\_RULE\_STATUS\_PARSING\_ERROR\_PHASE1\_CRYPTO = 0x0008000E,
21. FW\_RULE\_STATUS\_PARSING\_ERROR\_REMOTE\_ENDPOINTS = 0x0008000F,
22. FW\_RULE\_STATUS\_PARSING\_ERROR\_REMOTE\_ENDPOINT\_FQDN = 0x00080010,
23. FW\_RULE\_STATUS\_PARSING\_ERROR\_KEY\_MODULE = 0x00080011,
24. FW\_RULE\_STATUS\_PARSING\_ERROR\_LUA = 0x00080012,
25. FW\_RULE\_STATUS\_PARSING\_ERROR\_FWD\_LIFETIME = 0x00080013,
26. FW\_RULE\_STATUS\_PARSING\_ERROR\_TRANSPORT\_MACHINE\_AUTHZ\_SDDL = 0x00080014,
27. FW\_RULE\_STATUS\_PARSING\_ERROR\_TRANSPORT\_USER\_AUTHZ\_SDDL = 0x00080015,
28. FW\_RULE\_STATUS\_PARSING\_ERROR\_NETNAMES\_STRING = 0x00080016,
29. FW\_RULE\_STATUS\_PARSING\_ERROR\_SECURITY\_REALM\_ID\_STRING = 0x00080017,
30. FW\_RULE\_STATUS\_PARSING\_ERROR\_FQBN\_STRING = 0x00080018,
31. FW\_RULE\_STATUS\_SEMANTIC\_ERROR = 0x00100000,
32. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_RULE\_ID = 0x00100010,
33. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PORTS = 0x00100020,
34. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PORT\_KEYW = 0x00100021,
35. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PORT\_RANGE = 0x00100022,
36. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PORTRANGE\_RESTRICTION = 0x00100023,
37. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_V4\_SUBNETS = 0x00100040,
38. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_V6\_SUBNETS = 0x00100041,
39. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_V4\_RANGES = 0x00100042,
40. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_V6\_RANGES = 0x00100043,
41. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_RANGE = 0x00100044,
42. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_MASK = 0x00100045,
43. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_PREFIX = 0x00100046,
44. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_KEYW = 0x00100047,
45. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LADDR\_PROP = 0x00100048,
46. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_RADDR\_PROP = 0x00100049,
47. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_V6 = 0x0010004A,
48. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LADDR\_INTF = 0x0010004B,
49. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_V4 = 0x0010004C,
50. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TUNNEL\_ENDPOINT\_ADDR = 0x0010004D,
51. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DTE\_VER = 0x0010004E,
52. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DTE\_MISMATCH\_ADDR = 0x0010004F,
53. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PROFILE = 0x00100050,
54. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ICMP = 0x00100060,
55. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ICMP\_CODE = 0x00100061,
56. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_IF\_ID = 0x00100070,
57. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_IF\_TYPE = 0x00100071,
58. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ACTION = 0x00100080,
59. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ALLOW\_BYPASS = 0x00100081,
60. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DO\_NOT\_SECURE = 0x00100082,
61. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ACTION\_BLOCK\_IS\_ENCRYPTED\_SECURE = 0x00100083,
62. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_INCOMPATIBLE\_FLAG\_OR\_ACTION\_WITH\_SECURITY\_REALM = 0x00100084,
63. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DIR = 0x00100090,
64. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PROT = 0x001000A0,
65. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PROT\_PROP = 0x001000A1,
66. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DEFER\_EDGE\_PROP = 0x001000A2,
67. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ALLOW\_BYPASS\_OUTBOUND = 0x001000A3,
68. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DEFER\_USER\_INVALID\_RULE = 0x001000A4,
69. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS = 0x001000B0,
70. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTO\_AUTH = 0x001000B1,
71. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTO\_BLOCK = 0x001000B2,
72. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTO\_DYN\_RPC = 0x001000B3,
73. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTHENTICATE\_ENCRYPT = 0x001000B4,
74. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTH\_WITH\_ENC\_NEGOTIATE\_VER = 0x001000B5,
75. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTH\_WITH\_ENC\_NEGOTIATE = 0x001000B6,
76. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_ESP\_NO\_ENCAP\_VER = 0x001000B7,
77. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_ESP\_NO\_ENCAP = 0x001000B8,
78. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_TUNNEL\_AUTH\_MODES\_VER = 0x001000B9,
79. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_TUNNEL\_AUTH\_MODES = 0x001000BA,
80. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_IP\_TLS\_VER = 0x001000BB,
81. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_PORTRANGE\_VER = 0x001000BC,
82. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_ADDRS\_TRAVERSE\_DEFER\_VER = 0x001000BD,
83. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTH\_WITH\_ENC\_NEGOTIATE\_OUTBOUND = 0x001000BE,
84. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTHENTICATE\_WITH\_OUTBOUND\_BYPASS\_VER = 0x001000BF,
85. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_REMOTE\_AUTH\_LIST = 0x001000C0,
86. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_REMOTE\_USER\_LIST = 0x001000C1,
87. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LOCAL\_USER\_LIST = 0x001000C2,
88. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LUA\_VER = 0x001000C3,
89. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LOCAL\_USER\_OWNER = 0x001000C4,
90. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LOCAL\_USER\_OWNER\_VER = 0x001000C5,
91. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LUA\_CONDITIONAL\_VER = 0x001000C6,
92. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_SYSTEMOS\_GAMEOS = 0x001000C7,
93. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_CORTANA\_VER = 0x001000C8,
94. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_REMOTENAME = 0x001000C9
95. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_ALLOW\_PROFILE\_CROSSING\_VER = 0x001000D0,
96. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LOCAL\_ONLY\_MAPPED\_VER = 0x001000D1,
97. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PLATFORM = 0x001000E0,
98. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PLATFORM\_OP\_VER = 0x001000E1,
99. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PLATFORM\_OP = 0x001000E2,
100. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DTE\_NOANY\_ADDR = 0x001000F0,
101. FW\_RULE\_STATUS\_SEMANTIC\_TUNNEL\_EXEMPT\_WITH\_GATEWAY = 0x001000F1,
102. FW\_RULE\_STATUS\_SEMANTIC\_TUNNEL\_EXEMPT\_VER = 0x001000F2,
103. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_KEYWORD\_VER = 0x001000F3,
104. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_KEY\_MODULE\_VER = 0x001000F4,
105. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_APP\_CONTAINER\_PACKAGE\_ID = 0x00100100,
106. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_APP\_CONTAINER\_PACKAGE\_ID\_VER = 0x00100101,
107. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TRUST\_TUPLE\_KEYWORD\_INCOMPATIBLE = 0x00100200,
108. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TRUST\_TUPLE\_KEYWORD\_INVALID = 0x00100201,
109. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TRUST\_TUPLE\_KEYWORD\_VER = 0x00100202,
110. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_INTERFACE\_TYPES\_VER = 0x00100301,
111. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_NETNAMES\_VER = 0x00100401,
112. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_SECURITY\_REALM\_ID\_VER = 0x00100402,
113. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_SYSTEMOS\_GAMEOS\_VER = 0x00100403,
114. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DEVMODE\_VER = 0x00100404,
115. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_REMOTE\_SERVERNAME\_VER = 0x00100405,
116. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FQBN\_VER = 0x00100406,
117. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_COMPARTMENT\_ID\_VER = 0x00100407,
118. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_CALLOUT\_AND\_AUDIT\_VER = 0x00100408,
119. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_AUTH\_SET\_ID = 0x00100500,
120. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_SET\_ID = 0x00100510,
121. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_SET\_ID = 0x00100511,
122. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_KEY\_MANAGER\_DICTATE\_VER = 0x00100512,
123. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_KEY\_MANAGER\_NOTIFY\_VER = 0x00100513,
124. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TRANSPORT\_MACHINE\_AUTHZ\_VER = 0x00100514,
125. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TRANSPORT\_USER\_AUTHZ\_VER = 0x00100515,
126. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TRANSPORT\_MACHINE\_AUTHZ\_ON\_TUNNEL = 0x00100516,
127. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TRANSPORT\_USER\_AUTHZ\_ON\_TUNNEL = 0x00100517,
128. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PER\_RULE\_AND\_GLOBAL\_AUTHZ = 0x00100518,
129. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_SECURITY\_REALM = 0x00100519,
130. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_SET\_ID = 0x00101000,
131. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_IPSEC\_PHASE = 0x00101010,
132. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_EMPTY\_SUITES = 0x00101020,
133. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_AUTH\_METHOD = 0x00101030,
134. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_AUTH\_METHOD = 0x00101031,
135. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_METHOD\_ANONYMOUS = 0x00101032,
136. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_METHOD\_DUPLICATE = 0x00101033,
137. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_METHOD\_VER = 0x00101034,
138. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_SUITE\_FLAGS = 0x00101040,
139. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_HEALTH\_CERT = 0x00101041,
140. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_SIGNCERT\_VER = 0x00101042,
141. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_INTERMEDIATE\_CA\_VER = 0x00101043,
142. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_MACHINE\_SHKEY = 0x00101050,
143. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_CA\_NAME = 0x00101060,
144. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_MIXED\_CERTS = 0x00101061,
145. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_NON\_CONTIGUOUS\_CERTS = 0x00101062,
146. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_MIXED\_CA\_TYPE\_IN\_BLOCK = 0x00101063,
147. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_MACHINE\_USER\_AUTH = 0x00101070,
148. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_VER = 0x00101071,
149. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_VER\_MISMATCH = 0x00101072,
150. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_RENEWAL\_HASH = 0x00101073,
151. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_INVALID\_HASH = 0x00101074,
152. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_INVALID\_EKU = 0x00101075,
153. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_INVALID\_NAME\_TYPE = 0x00101076,
154. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_INVALID\_NAME = 0x00101077,
155. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_INVALID\_CRITERIA\_TYPE = 0x00101078,
156. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_MISSING\_CRITERIA = 0x00101079,
157. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PROXY\_SERVER = 0x00101080,
158. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_PROXY\_SERVER\_VER = 0x00101081,
159. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_NON\_DEFAULT\_ID = 0x00105000,
160. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_FLAGS = 0x00105001,
161. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_TIMEOUT\_MINUTES = 0x00105002,
162. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_TIMEOUT\_SESSIONS = 0x00105003,
163. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_KEY\_EXCHANGE = 0x00105004,
164. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_ENCRYPTION = 0x00105005,
165. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_HASH = 0x00105006,
166. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_ENCRYPTION\_VER = 0x00105007,
167. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_HASH\_VER = 0x00105008,
168. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_KEY\_EXCH\_VER = 0x00105009,
169. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_PFS = 0x00105020,
170. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_PROTOCOL = 0x00105021,
171. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_ENCRYPTION = 0x00105022,
172. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_HASH = 0x00105023,
173. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_TIMEOUT\_MINUTES = 0x00105024,
174. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_TIMEOUT\_KBYTES = 0x00105025,
175. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_ENCRYPTION\_VER = 0x00105026,
176. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_HASH\_VER = 0x00105027,
177. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_PFS\_VER = 0x00105028,
178. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_CRYPTO\_ENCR\_HASH = 0x00105040,
179. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_CRYPTO\_ENCR\_HASH\_COMPAT = 0x00105041,
180. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_SCHEMA\_VERSION = 0x00105050,
181. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_OR\_AND\_CONDITIONS = 0x00106000,
182. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_AND\_CONDITIONS = 0x00106001,
183. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_CONDITION\_KEY = 0x00106002,
184. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_CONDITION\_MATCH\_TYPE = 0x00106003,
185. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_CONDITION\_DATA\_TYPE = 0x00106004,
186. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_CONDITION\_KEY\_AND\_DATA\_TYPE = 0x00106005,
187. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEYS\_PROTOCOL\_PORT = 0x00106006,
188. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_PROFILE = 0x00106007,
189. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_STATUS = 0x00106008,
190. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_FILTERID = 0x00106009,
191. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_APP\_PATH = 0x00106010,
192. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_PROTOCOL = 0x00106011,
193. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_LOCAL\_PORT = 0x00106012,
194. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_REMOTE\_PORT = 0x00106013,
195. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_SVC\_NAME = 0x00106015,
196. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_REQUIRE\_IN\_CLEAR\_OUT\_ON\_TRANSPORT = 0x00107000,
197. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TUNNEL\_BYPASS\_TUNNEL\_IF\_SECURE\_ON\_TRANSPORT = 0x00107001,
198. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_NOENCAP\_ON\_TUNNEL = 0x00107002,
199. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_NOENCAP\_ON\_PSK = 0x00107003,
200. FW\_RULE\_STATUS\_RUNTIME\_ERROR = 0x00200000,
201. FW\_RULE\_STATUS\_RUNTIME\_ERROR\_PHASE1\_AUTH\_NOT\_FOUND = 0x00200001,
202. FW\_RULE\_STATUS\_RUNTIME\_ERROR\_PHASE2\_AUTH\_NOT\_FOUND = 0x00200002,
203. FW\_RULE\_STATUS\_RUNTIME\_ERROR\_PHASE2\_CRYPTO\_NOT\_FOUND = 0x00200003,
204. FW\_RULE\_STATUS\_RUNTIME\_ERROR\_AUTH\_MCHN\_SHKEY\_MISMATCH = 0x00200004,
205. FW\_RULE\_STATUS\_RUNTIME\_ERROR\_PHASE1\_CRYPTO\_NOT\_FOUND = 0x00200005,
206. FW\_RULE\_STATUS\_RUNTIME\_ERROR\_AUTH\_NOENCAP\_ON\_TUNNEL = 0x00200006,
207. FW\_RULE\_STATUS\_RUNTIME\_ERROR\_AUTH\_NOENCAP\_ON\_PSK = 0x00200007,
208. FW\_RULE\_STATUS\_RUNTIME\_ERROR\_KEY\_MODULE\_AUTH\_MISMATCH = 0x00200008,
209. FW\_RULE\_STATUS\_ERROR = FW\_RULE\_STATUS\_PARSING\_ERROR | FW\_RULE\_STATUS\_SEMANTIC\_ERROR | FW\_RULE\_STATUS\_RUNTIME\_ERROR,
210. FW\_RULE\_STATUS\_ALL = 0xFFFF0000
211. } FW\_RULE\_STATUS;

**FW\_RULE\_STATUS\_OK:** The rule was parsed successfully from the store, is correctly constructed, and has no issue.

**FW\_RULE\_STATUS\_PARTIALLY\_IGNORED:** The rule has fields that the service can successfully ignore. The ignored fields can be present only if the policy (such as the [**Group Policy**](#gt_defe8c22-1365-4e5e-abf7-46ad112d3bda)) was written by future firewall and advanced security components that support a higher schema version. Therefore, this error occurs only if the version of the rule is higher; specifically, a higher minor version means that part of the rule might not be understandable. Because the host firewall component does not understand these new fields, it cannot meaningfully specify what was ignored in the rule.

**FW\_RULE\_STATUS\_IGNORED:** The rule has a higher major version that the service MUST ignore. Higher major schema versions specify that nothing in the rule is understandable to lower major version components.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_NAME:** The name contains characters that are not valid or the length is not valid.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_DESC:** The description contains characters that are not valid or the length is not valid.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_APP:** The application contains characters that are not valid or the length is not valid.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_SVC:** The service contains characters that are not valid or the length is not valid.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_RMA:** The remote machine authentication contains characters that are not valid or the length is not valid.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_RUA:** The remote user authentication contains characters that are not valid or the length is not valid.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_EMBD:** The embedded context contains characters that are not valid or the length is not valid.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_RULE\_ID:** The rule ID contains characters that are not valid or the length is not valid.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_PHASE1\_AUTH:** The Phase1 authentication set ID contains characters that are not valid or the length is not valid.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_PHASE2\_CRYPTO:** The Phase2 cryptographic set ID contains characters that are not valid or the length is not valid.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_PHASE2\_AUTH:** The Phase2 authentication set ID contains characters that are not valid or the length is not valid.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_RESOLVE\_APP:** The application name cannot be resolved.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_MAINMODE\_ID:** This error is unused and not returned by the system.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_PHASE1\_CRYPTO:** The Phase1 cryptographic set ID contains characters that are not valid or the length is not valid.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_REMOTE\_ENDPOINTS:** The remote tunnel [**endpoints**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee) contain characters that are not valid, or the length is not valid.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_REMOTE\_ENDPOINT\_FQDN:** The remote tunnel endpoint [**fully qualified domain name (FQDN)**](#gt_1769aec9-237e-44ed-9014-1abb3ec6de6e) contains characters that are not valid, or the length is not valid.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_KEY\_MODULE:** The keying modules contain characters that are not valid, or the length is not valid.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_LUA:** The local user authorization list contains characters that are not valid or the length is not valid.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_FWD\_LIFETIME:** The forward path [**security association (SA)**](#gt_67cbf867-7a49-41f3-a68f-37b5f9035acb) lifetime contains characters that are not valid or the length is not valid.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_TRANSPORT\_MACHINE\_AUTHZ\_SDDL:** The [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) transport mode machine authorization SDDL string contains characters that are not valid, or the length is not valid.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_TRANSPORT\_USER\_AUTHZ\_SDDL:** The IPsec transport mode user authorization SDDL string contains characters that are not valid, or the length is not valid.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_NETNAMES\_STRING**: A string for the network name structure is invalid.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_SECURITY\_REALM\_ID\_STRING**: A string for the security realm ID is invalid.

**FW\_RULE\_STATUS\_PARSING\_ERROR\_FQBN\_STRING**: A string for the [**fully qualified binary name (FQBN)**](#gt_bbf47ea1-11e7-447c-848d-5a1277648312) is invalid; also see [[MSDN-FQBN]](https://go.microsoft.com/fwlink/?linkid=839018).

**FW\_RULE\_STATUS\_PARSING\_ERROR:** The rule did not parse correctly.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_RULE\_ID:** Semantic error: The rule ID is not specified.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PORTS:** Semantic error: Mismatch in the number of ports and port buffers.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PORT\_KEYW:** Semantic error: The port keyword is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PORT\_RANGE:** Semantic error: End != Begin or port = 0.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PORTRANGE\_RESTRICTION:** Semantic error: A port range has been specified for a connection security rule, but the action is not Do Not Secure.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_V4\_SUBNETS:** Semantic error: Mismatch in the number of v4 subnets and subnet buffers.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_V6\_SUBNETS:** Semantic error: Mismatch in the number of v6 subnets and subnet buffers.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_V4\_RANGES:** Semantic error: Mismatch in the number of v4 ranges and range buffers.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_V6\_RANGES:** Semantic error: Mismatch in the number of v6 ranges and range buffers.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_RANGE:** Semantic error: End < Begin.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_MASK:** Semantic error: The mask specified on a v4 subnet is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_PREFIX:** Semantic error: The prefix specified on a v6 subnet is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_KEYW:** Semantic error: The specified keyword is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LADDR\_PROP:** Semantic error: A property on local addresses does not belong to the LocalAddress.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_RADDR\_PROP:** Semantic error: A property on remote addresses does not belong to the RemoteAddress.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_V6:** Semantic error: An unspecified or loopback IPv6 address was specified.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LADDR\_INTF:** Semantic error: A local address cannot be used together with either an interface or an interface type.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_V4:** Semantic error: An unspecified or loopback IPv4 address was specified.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TUNNEL\_ENDPOINT\_ADDR:** Semantic error: An endpoint "any" cannot be specified for a tunnel mode rule.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DTE\_VER:** Semantic error: An incorrect schema version was specified for using dynamic tunnel endpoints.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DTE\_MISMATCH\_ADDR:** Semantic error: The v4 and v6 tunnel endpoints are neither local nor remote endpoints.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PROFILE:** Semantic error: The profile type is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ICMP:** Semantic error: Mismatch in the number of ICMPs and ICMP buffers.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ICMP\_CODE:** Semantic error: The specified ICMP code is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_IF\_ID:** Semantic error: Mismatch in the number of interfaces and interface buffers.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_IF\_TYPE:** Semantic error: The specified interface type is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ACTION:** Semantic error: The specified action is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ALLOW\_BYPASS:** Semantic error: An allow-bypass action is specified, but the rule does not meet allow-bypass criteria (such as, the direction is inbound, authenticate/encrypt flags are set, or remote machine authentication is set).

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DO\_NOT\_SECURE:** Semantic error: A DO\_NOT\_SECURE action is specified together with authentication or cryptographic sets.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ACTION\_BLOCK\_IS\_ENCRYPTED\_SECURE:** Semantic error: A block action was specified together with a require security or a require encryption action.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DIR:** Semantic error: The specified direction is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PROT:** Semantic error: The specified protocol is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PROT\_PROP:** Semantic error: The protocol and protocol-dependent fields do not match.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DEFER\_EDGE\_PROP:** Semantic error: A Dynamic edge flag (either defer to app or defer to user) is set without having an edge flag set.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ALLOW\_BYPASS\_OUTBOUND:** Semantic error: An outbound allow-bypass action is specified, but the rule does not meet allow-bypass criteria (authenticate/encrypt flags set).

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DEFER\_USER\_INVALID\_RULE:** The rule does not allow the defer user property to be set.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS:** Semantic error: The specified flags are not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTO\_AUTH:** Semantic error: The autogenerate flag is set, but no authentication flags are set.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTO\_BLOCK:** Semantic error: The autogenerate flag is set, but the action is block.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTO\_DYN\_RPC:** Semantic error: The autogenerate flag is set together with the dynamic [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) flag.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTHENTICATE\_ENCRYPT:** Semantic error: The authenticate and authenticate-encrypt flags are both specified.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTH\_WITH\_ENC\_NEGOTIATE\_VER:** Semantic error: The schema version is not compliant with the Authenticate with Encryption flag.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTH\_WITH\_ENC\_NEGOTIATE:** Semantic error: The Authenticate with Encryption Negotiate flag is specified but the basic Authenticate with Encryption flag is not set.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_ESP\_NO\_ENCAP\_VER:** Semantic error: The schema version is not compliant with the Authenticate with No Encapsulation flag.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_ESP\_NO\_ENCAP:** Semantic error: The Authenticate with No Encapsulation flag is specified but the basic Authenticate flag is not set.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_TUNNEL\_AUTH\_MODES\_VER:** Semantic error: The schema version is not compliant with the tunnel authentication modes.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_TUNNEL\_AUTH\_MODES:** Semantic error: The tunnel authentication modes are specified by a lower-version client.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_IP\_TLS\_VER:** Semantic error: The schema version is not compliant with the IP\_TLS flag.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_PORTRANGE\_VER:** Semantic error: The schema version is not compliant with port range support.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_ADDRS\_TRAVERSE\_DEFER\_VER:** Semantic error: The schema version is not compliant with the FW\_RULE\_FLAGS\_ROUTEABLE\_ADDRS\_TRAVERSE\_DEFER\_APP flag. For more information, see [2.2.34](#Section_fc67ea0419f04ccd8912abe467e7c11a).

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTH\_WITH\_ENC\_NEGOTIATE\_OUTBOUND:** Semantic error: The Authenticate with Encryption Negotiate flag is set for the outbound rule.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTHENTICATE\_WITH\_OUTBOUND\_BYPASS\_VER:** Semantic error: The Outbound Authenticated bypass is not supported on this version.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_REMOTE\_AUTH\_LIST:** Semantic error: An authorized remote machine or user list is specified, but the authenticate/encryption flags were not set.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_REMOTE\_USER\_LIST:** Semantic error: An authorized remote user list is specified on an outbound direction.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LOCAL\_USER\_LIST:** Semantic error: The authorized local user list is specified, but a local service has also been specified.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LUA\_VER:** Semantic error: The schema version is not compliant with the authorized local user list.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LOCAL\_USER\_OWNER:** Semantic error: The local user owner is specified, but a local service has also been specified.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LOCAL\_USER\_OWNER\_VER:** Semantic error: The schema version is not compliant with the local user owner.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_ALLOW\_PROFILE\_CROSSING\_VER:** Semantic error: The schema version is not compliant with profile crossing.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LOCAL\_ONLY\_MAPPED\_VER:** Semantic error: The schema version is not compliant with local-only mappings.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PLATFORM:** Semantic error: The number of valid operating system platforms and the list of valid operating system platforms do not match.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PLATFORM\_OP\_VER:** Semantic error: Schema version not compliant with the platform operator used.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PLATFORM\_OP:** Semantic error: Invalid platform operator used.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DTE\_NOANY\_ADDR:** Semantic error: DTE is specified but all tunnel endpoints are specified.

**FW\_RULE\_STATUS\_SEMANTIC\_TUNNEL\_EXEMPT\_WITH\_GATEWAY:** Semantic error: DTM tunnel exemption specified with tunnel endpoint (gateways) address.

**FW\_RULE\_STATUS\_SEMANTIC\_TUNNEL\_EXEMPT\_VER:** Semantic error: Schema version not compliant with tunnel mode exemptions.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_KEYWORD\_VER:** Semantic error: The schema version is not compliant with one or more address keywords.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_KEY\_MODULE\_VER:** Semantic error: The schema version is not compliant with the keying modules.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_APP\_CONTAINER\_PACKAGE\_ID:** Semantic error: The application container package ID is not a valid [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d).

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_APP\_CONTAINER\_PACKAGE\_ID\_VER:** Semantic error: The schema version is not compliant with application containers.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TRUST\_TUPLE\_KEYWORD\_INCOMPATIBLE:** Semantic error: Trust tuple keywords are specified, but specific addresses or ports have also been specified.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TRUST\_TUPLE\_KEYWORD\_INVALID:** Semantic error: One or more trust tuple keywords is invalid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TRUST\_TUPLE\_KEYWORD\_VER:** Semantic error: The schema version is not compliant with the trust tuple keywords.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_AUTH\_SET\_ID:** Semantic error: Phase1 authentication set ID is not specified.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_SET\_ID:** Semantic error: Phase2 cryptographic set ID is not specified.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_SET\_ID:** Semantic error: Phase1 cryptographic set ID is not specified.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_KEY\_MANAGER\_DICTATE\_VER:** Semantic error: The schema version is not compliant with the Key Manager Dictation flag.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_KEY\_MANAGER\_NOTIFY\_VER:** Semantic error: The schema version is not compliant with the Key Manager Notification flag.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TRANSPORT\_MACHINE\_AUTHZ\_VER:** Semantic error: The schema version is not compliant with IPsec transport mode machine authorization lists.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TRANSPORT\_USER\_AUTHZ\_VER:** Semantic error: The schema version is not compliant with IPsec transport mode user authorization lists.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TRANSPORT\_MACHINE\_AUTHZ\_ON\_TUNNEL:** Semantic error: An IPsec transport mode machine authorization list is specified on a tunnel mode rule.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TRANSPORT\_USER\_AUTHZ\_ON\_TUNNEL:** Semantic error: An IPsec transport mode user authorization list is specified on a tunnel mode rule.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PER\_RULE\_AND\_GLOBAL\_AUTHZ:** Semantic error: The Apply Global Authorization flag is set, but a per-rule authorization list is also specified.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_SET\_ID:** Semantic error: The set ID is not specified.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_IPSEC\_PHASE:** Semantic error: The specified phase is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_EMPTY\_SUITES:** Semantic error: No suites are specified in the set.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_AUTH\_METHOD:** Semantic error: The Phase1 authentication method is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_AUTH\_METHOD:** Semantic error: The Phase2 authentication method is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_METHOD\_ANONYMOUS:** Semantic error: Anonymous authentication is specified as the only authentication proposal (or authentication proposal suite).

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_METHOD\_DUPLICATE:** Semantic error: Duplicate authentication methods are specified but not supported.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_METHOD\_VER:** Semantic error: Suite specifies authentication method that is not compliant with its schema version.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_SUITE\_FLAGS:** Semantic error: The specified authentication suite flags are not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_HEALTH\_CERT:** Semantic error: The machine certificate MUST be a health certificate for Phase2 authentication.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_SIGNCERT\_VER:** Semantic error: The suite specifies signing that is not compliant with its schema version.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_INTERMEDIATE\_CA\_VER:** Semantic error: Specifies an intermediate [**certificate authority (CA)**](#gt_c925d5d7-a442-4ba4-9586-5f94ccec847a) that is not compliant with its schema version.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_MACHINE\_SHKEY:** Semantic error: The machine shared key is either missing or not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_CA\_NAME:** Semantic error: The CA name is either missing or not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_MIXED\_CERTS:** Semantic error: Health certificates (CERTS) cannot be specified together with regular certificates.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_NON\_CONTIGUOUS\_CERTS:** Semantic error: Certificates that have a specific signing algorithm are not contiguous.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_MIXED\_CA\_TYPE\_IN\_BLOCK:** Semantic error: Both root and intermediate CA types cannot be present in the same signing algorithm block.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_MACHINE\_USER\_AUTH:** Semantic error: Both machine and user authentications are specified.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_VER:** The suite specifies certificate criteria but the schema version does not allow certificate criteria to be present. Certificate criteria are supported only in schemas with version number 2.20 and greater.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_VER\_MISMATCH:** The version specified for the criteria structure is different from the auth set version.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_RENEWAL\_HASH:** Cert criteria were specified for a non-cert authentication method.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_INVALID\_HASH:** An invalid hash was specified in the criteria. A valid hash is a string of hex characters (40 characters in length).

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_INVALID\_EKU:** An invalid [**EKU**](#gt_06beeb29-6e93-4472-a53d-bbd51eca5759) was specified. Validity checking of an EKU involves checking that the EKU is composed of characters representing 0 to 9 and ".".

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_INVALID\_NAME\_TYPE:** A name type greater than FW\_CERT\_CRITERIA\_NAME\_MAX was specified.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_INVALID\_NAME:** A name type was specified but either a NULL name is also specified, or the number of characters in the name is greater than FW\_MAX\_RULE\_STRING\_LEN(10000), or the name string contains the "|" character.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_INVALID\_CRITERIA\_TYPE:** The criteria type specified is greater than FW\_CERT\_CRITERIA\_TYPE\_MAX.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_MISSING\_CRITERIA:** The specified suites are missing either selection or validation criteria.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PROXY\_SERVER:** Semantic error: The [**Kerberos**](#gt_d6a282ce-b1da-41e1-b05a-22f777a5c1fe) proxy server name is not a valid fully qualified domain name (FQDN).

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_PROXY\_SERVER\_VER:** Semantic error: The schema version is not compliant with Kerberos proxy servers.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_NON\_DEFAULT\_ID:** Semantic error: The ID for the Phase1 cryptographic set is not the default.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_FLAGS:** Semantic error: The Phase1 cryptographic set flags are not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_TIMEOUT\_MINUTES:** Semantic error: The Phase1 cryptographic set time-out minutes are not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_TIMEOUT\_SESSIONS:** Semantic error: The time-out sessions for the Phase1 cryptographic set are not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_KEY\_EXCHANGE:** Semantic error: The key exchange for the Phase1 cryptographic set is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_ENCRYPTION:** Semantic error: The Phase1 cryptographic set encryption is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_HASH:** Semantic error: The Phase1 cryptographic set hash is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_ENCRYPTION\_VER:** Semantic error: The Phase1 cryptographic set encryption is not schema-version compliant.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_HASH\_VER:** Semantic error: The Phase1 cryptographic set hash is not schema version compliant.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_KEY\_EXCH\_VER:** Semantic error: The schema version is not compliant with one or more of the specified main mode key exchange algorithms.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_PFS:** Semantic error: The Phase2 cryptographic set [**perfect forward secrecy (PFS)**](#gt_5d8948bc-5e32-483b-906d-42f785d0df18) is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_PROTOCOL:** Semantic error: The Phase2 cryptographic set protocol is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_ENCRYPTION:** Semantic error: The Phase2 cryptographic set encryption is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_HASH:** Semantic error: The Phase2 cryptographic set hash is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_TIMEOUT\_MINUTES:** Semantic error: The Phase2 cryptographic set time-out minutes are not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_TIMEOUT\_KBYTES:** Semantic error: The Phase2 cryptographic set time-out kilobytes are not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_ENCRYPTION\_VER:** Semantic error: The Phase2 cryptographic set encryption is not schema-version compliant.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_HASH\_VER:** The Phase2 cryptographic set hash is not schema-version compliant.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_PFS\_VER:** Semantic error: The schema version is not compliant with the specified Phase2 perfect forward secrecy (PFS) option.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_CRYPTO\_ENCR\_HASH:** Semantic error: Neither the encryption nor the hash is specified.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_CRYPTO\_ENCR\_HASH\_COMPAT:** Semantic error: The encryption and hash use incompatible algorithms.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_SCHEMA\_VERSION:** Semantic error: The specified schema version is lower than the lowest supported version.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_OR\_AND\_CONDITIONS:** Semantic error: A mismatch exists in the number of OR'd terms and term arrays.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_AND\_CONDITIONS:** Semantic error: A mismatch exists in the number of AND'd conditions and condition arrays.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_CONDITION\_KEY:** Semantic error: The condition match key is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_CONDITION\_MATCH\_TYPE:** Semantic error: The condition match type is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_CONDITION\_DATA\_TYPE:** Semantic error: The condition data type is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_CONDITION\_KEY\_AND\_DATA\_TYPE:** Semantic error: The key and data type combination is not valid.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEYS\_PROTOCOL\_PORT:** Semantic error: A port condition is present without a protocol condition.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_PROFILE:** Semantic error: The profile key is unavailable for the queried object type.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_STATUS:** Semantic error: The status key is unavailable for the queried object type.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_FILTERID:** Semantic error: The FilterID key is unavailable for the queried object type.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_APP\_PATH:** Semantic error: The application key is unavailable for the queried object type.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_PROTOCOL:** Semantic error: The protocol key is unavailable for the queried object type.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_LOCAL\_PORT:** Semantic error: The local port key is unavailable for the queried object type.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_REMOTE\_PORT:** Semantic error: The remote port key is unavailable for the queried object type.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_SVC\_NAME:** Semantic error: The service name key is unavailable for the queried object type.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_REQUIRE\_IN\_CLEAR\_OUT\_ON\_TRANSPORT:** Semantic error: "Require in clear out" tunnel authentication mode cannot be set on transport mode rules.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TUNNEL\_BYPASS\_TUNNEL\_IF\_SECURE\_ON\_TRANSPORT:** Semantic error: Cannot set flag to exempt IPsec transport traffic from a tunnel mode, on a transport rule.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_NOENCAP\_ON\_TUNNEL:** Semantic error: Cannot set FW\_CRYPTO\_PROTOCOL\_AUTH\_NO\_ENCAP (see section [2.2.68](#Section_d97bac3603e842159984f2fbddd66be0)) on a tunnel mode rule.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_NOENCAP\_ON\_PSK:** Semantic error: Cannot mix FW\_CRYPTO\_PROTOCOL\_AUTH\_NO\_ENCAP (see section 2.2.68) protocol with Preshared key authentication methods.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_CRYPTO\_ENCR\_HASH:** Semantic error: Both the encryption and hash are not specified.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_CRYPTO\_ENCR\_HASH\_COMPAT:** Semantic error: The encryption and hash use incompatible algorithms.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_SCHEMA\_VERSION:** Semantic error: The specified schema version is earlier than the supported versions.

**FW\_RULE\_STATUS\_SEMANTIC\_ERROR:** There is a semantic error when considering the fields of the rule in conjunction with other policy objects.

**FW\_RULE\_STATUS\_RUNTIME\_ERROR\_PHASE1\_AUTH\_NOT\_FOUND:** A Phase1 authentication set is not found.

**FW\_RULE\_STATUS\_RUNTIME\_ERROR\_PHASE2\_AUTH\_NOT\_FOUND:** A Phase2 authentication set is not found.

**FW\_RULE\_STATUS\_RUNTIME\_ERROR\_PHASE2\_CRYPTO\_NOT\_FOUND:** A Phase2 cryptographic set is not found.

**FW\_RULE\_STATUS\_RUNTIME\_ERROR\_AUTH\_MCHN\_SHKEY\_MISMATCH:** A Phase2 authentication set cannot be specified when the Phase1 authentication set contains a pre-shared key as an authentication method.

**FW\_RULE\_STATUS\_RUNTIME\_ERROR\_PHASE1\_CRYPTO\_NOT\_FOUND:** A Phase1 cryptographic set is not found.

**FW\_RULE\_STATUS\_RUNTIME\_ERROR\_AUTH\_NOENCAP\_ON\_TUNNEL:** Semantic error: Cannot set FW\_CRYPTO\_PROTOCOL\_AUTH\_NO\_ENCAP (see section 2.2.68) on a tunnel mode rule.

**FW\_RULE\_STATUS\_RUNTIME\_ERROR\_AUTH\_NOENCAP\_ON\_PSK:** Semantic error: Cannot mix FW\_CRYPTO\_PROTOCOL\_AUTH\_NO\_ENCAP (see section 2.2.68) protocol with Preshared key authentication methods.

**FW\_RULE\_STATUS\_RUNTIME\_ERROR\_KEY\_MODULE\_AUTH\_MISMATCH:** Semantic error: The key module in the rule is incompatible with the authentication methods specified in the associated authentication sets.

**FW\_RULE\_STATUS\_RUNTIME\_ERROR:** There is a runtime error when the object is considered with other policy objects.

**FW\_RULE\_STATUS\_ERROR:** An error of any kind occurred. This symbolic constant has a value of 0x00380000.

**FW\_RULE\_STATUS\_ALL:** The status of all (it is used to enumerate all the rules, regardless of the status).

### FW\_RULE\_STATUS\_CLASS

This enumeration defines classes of status codes.

1. typedef enum \_tag\_FW\_RULE\_STATUS\_CLASS
2. {
3. FW\_RULE\_STATUS\_CLASS\_OK = FW\_RULE\_STATUS\_OK,
4. FW\_RULE\_STATUS\_CLASS\_PARTIALLY\_IGNORED = FW\_RULE\_STATUS\_PARTIALLY\_IGNORED,
5. FW\_RULE\_STATUS\_CLASS\_IGNORED = FW\_RULE\_STATUS\_IGNORED,
6. FW\_RULE\_STATUS\_CLASS\_PARSING\_ERROR = FW\_RULE\_STATUS\_PARSING\_ERROR,
7. FW\_RULE\_STATUS\_CLASS\_SEMANTIC\_ERROR = FW\_RULE\_STATUS\_SEMANTIC\_ERROR,
8. FW\_RULE\_STATUS\_CLASS\_RUNTIME\_ERROR = FW\_RULE\_STATUS\_RUNTIME\_ERROR,
9. FW\_RULE\_STATUS\_CLASS\_ERROR = FW\_RULE\_STATUS\_ERROR,
10. FW\_RULE\_STATUS\_CLASS\_ALL = FW\_RULE\_STATUS\_ALL
11. } FW\_RULE\_STATUS\_CLASS;

**FW\_RULE\_STATUS\_CLASS\_OK**: The rule is correctly constructed and has no issue. This symbolic constant has a value of 0x00010000.

**FW\_RULE\_STATUS\_CLASS\_PARTIALLY\_IGNORED**: The rule has fields that the service can successfully ignore. This symbolic constant has a value of 0x00020000.

**FW\_RULE\_STATUS\_CLASS\_IGNORED**: The rule has a higher version that the service MUST ignore. This symbolic constant has a value of 0x00040000.

**FW\_RULE\_STATUS\_CLASS\_PARSING\_ERROR**: The rule failed to be parsed correctly. This symbolic constant has a value of 0x00080000.

**FW\_RULE\_STATUS\_CLASS\_SEMANTIC\_ERROR**: There is a semantic error when considering the fields of the rule in conjunction. This symbolic constant has a value of 0x00100000.

**FW\_RULE\_STATUS\_CLASS\_RUNTIME\_ERROR**: There is a runtime error when the object is considered in conjunction with other policy objects. This symbolic constant has a value of 0x00200000.

**FW\_RULE\_STATUS\_CLASS\_ERROR**: An error occurred. This symbolic constant has a value of 0x00380000.

**FW\_RULE\_STATUS\_CLASS\_ALL**: The status of all (used to enumerate ALL the rules, regardless of the status). This symbolic constant has a value of 0xFFFF0000.

### FW\_OBJECT\_CTRL\_FLAG

This enumeration is used to indicate the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) protocol when elements in structures are included.

1. typedef enum \_tag\_FW\_OBJECT\_CTRL\_FLAG
2. {
3. FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA = 0x0001
4. } FW\_OBJECT\_CTRL\_FLAG;

**FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA:** This flag indicates that the structure where this flag is specified contains metadata information.

### FW\_ENFORCEMENT\_STATE

This enumeration is part of the metadata information. It provides information about whether or not the policy expressed by an object is currently being enforced by the server.

1. typedef enum \_tag\_FW\_ENFORCEMENT\_STA
2. {
3. FW\_ENFORCEMENT\_STATE\_INVALID = 0,
4. FW\_ENFORCEMENT\_STATE\_FULL = 1,
5. FW\_ENFORCEMENT\_STATE\_WF\_OFF\_IN\_PROFILE = 2,
6. FW\_ENFORCEMENT\_STATE\_CATEGORY\_OFF = 3,
7. FW\_ENFORCEMENT\_STATE\_DISABLED\_OBJECT = 4,
8. FW\_ENFORCEMENT\_STATE\_INACTIVE\_PROFILE = 5,
9. FW\_ENFORCEMENT\_STATE\_LOCAL\_ADDRESS\_RESOLUTION\_EMPTY = 6,
10. FW\_ENFORCEMENT\_STATE\_REMOTE\_ADDRESS\_RESOLUTION\_EMPTY = 7,
11. FW\_ENFORCEMENT\_STATE\_LOCAL\_PORT\_RESOLUTION\_EMPTY = 8,
12. FW\_ENFORCEMENT\_STATE\_REMOTE\_PORT\_RESOLUTION\_EMPTY = 9,
13. FW\_ENFORCEMENT\_STATE\_INTERFACE\_RESOLUTION\_EMPTY = 10,
14. FW\_ENFORCEMENT\_STATE\_APPLICATION\_RESOLUTION\_EMPTY = 12,
15. FW\_ENFORCEMENT\_STATE\_REMOTE\_MACHINE\_EMPTY = 12,
16. FW\_ENFORCEMENT\_STATE\_REMOTE\_USER\_EMPTY = 13,
17. FW\_ENFORCEMENT\_STATE\_LOCAL\_GLOBAL\_OPEN\_PORTS\_DISALLOWED = 14,
18. FW\_ENFORCEMENT\_STATE\_LOCAL\_AUTHORIZED\_APPLICATIONS\_DISALLOWED = 15,
19. FW\_ENFORCEMENT\_STATE\_LOCAL\_FIREWALL\_RULES\_DISALLOWED = 16,
20. FW\_ENFORCEMENT\_STATE\_LOCAL\_CONSEC\_RULES\_DISALLOWED = 17,
21. FW\_ENFORCEMENT\_STATE\_MISMATCHED\_PLATFORM = 18,
22. FW\_ENFORCEMENT\_STATE\_OPTIMIZED\_OUT = 19,
23. FW\_ENFORCEMENT\_STATE\_LOCAL\_USER\_EMPTY = 20,
24. FW\_ENFORCEMENT\_STATE\_TRANSPORT\_MACHINE\_SD\_EMPTY = 21,
25. FW\_ENFORCEMENT\_STATE\_TRANSPORT\_USER\_SD\_EMPTY = 22,
26. FW\_ENFORCEMENT\_STATE\_TUPLE\_RESOLUTION\_EMPTY = 23,
27. FW\_ENFORCEMENT\_STATE\_MAX = 24
28. } FW\_ENFORCEMENT\_STATE;

**FW\_ENFORCEMENT\_STATE\_INVALID:** This value is invalid and MUST NOT be used by the server. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 0.

**FW\_ENFORCEMENT\_STATE\_FULL:** The object is being enforced. This symbolic constant has a value of 1.

**FW\_ENFORCEMENT\_STATE\_WF\_OFF\_IN\_PROFILE:** The object is not being enforced because the firewall and advanced security component is not active in a profile where the object is meant to be applied. This symbolic constant has a value of 2.

**FW\_ENFORCEMENT\_STATE\_CATEGORY\_OFF:** The object is not being enforced because a third-party software component registered with the firewall and advanced security component to own the functionality that the object is meant to perform. This symbolic constant has a value of 3.

**FW\_ENFORCEMENT\_STATE\_DISABLED\_OBJECT:** The object is not being enforced because the object is disabled. This symbolic constant has a value of 4.

**FW\_ENFORCEMENT\_STATE\_INACTIVE\_PROFILE:** The object is not being enforced because at least one of the profiles that the object is meant to be applied to is not currently active. This symbolic constant has a value of 5.

**FW\_ENFORCEMENT\_STATE\_LOCAL\_ADDRESS\_RESOLUTION\_EMPTY:** The object is not being enforced because the local address condition of the object contains a keyword that resolves to an empty set. This symbolic constant has a value of 6.

**FW\_ENFORCEMENT\_STATE\_REMOTE\_ADDRESS\_RESOLUTION\_EMPTY:** The object is not being enforced because the remote address condition of the object contains a keyword that resolves to an empty set. This symbolic constant has a value of 7.

**FW\_ENFORCEMENT\_STATE\_LOCAL\_PORT\_RESOLUTION\_EMPTY:** The object is not being enforced because the local port condition of the object contains a keyword that resolves to an empty set. This symbolic constant has a value of 8.

**FW\_ENFORCEMENT\_STATE\_REMOTE\_PORT\_RESOLUTION\_EMPTY:** The object is not being enforced because the remote port condition of the object contains a keyword that resolves to an empty set. This symbolic constant has a value of 9.

**FW\_ENFORCEMENT\_STATE\_INTERFACE\_RESOLUTION\_EMPTY:** The object is not being enforced because the interface condition of the object contains a keyword that resolves to an empty set. This symbolic constant has a value of 10.

**FW\_ENFORCEMENT\_STATE\_APPLICATION\_RESOLUTION\_EMPTY:** The object is not being enforced because the application condition of the object contains a path that could not resolve to a valid file system path. This symbolic constant has a value of 11.

**FW\_ENFORCEMENT\_STATE\_REMOTE\_MACHINE\_EMPTY:** The object is not being enforced because the remote machine condition of the object contains an SDDL with a [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) that is not currently available on the host. This symbolic constant has a value of 12.

**FW\_ENFORCEMENT\_STATE\_REMOTE\_USER\_EMPTY:** The object is not being enforced because the remote user condition of the object contains an SDDL with a SID that is not currently available on the host. This symbolic constant has a value of 13.

**FW\_ENFORCEMENT\_STATE\_LOCAL\_GLOBAL\_OPEN\_PORTS\_DISALLOWED:** The object is not being enforced because the FW\_PROFILE\_CONFIG\_AUTH\_APPS\_ALLOW\_USER\_PREF\_MERGE configuration option (see section [2.2.37](#Section_5a6e0d39802d456bb483c7360566fcdd) for more details) from a profile that the object applied to, disallowed its use. This symbolic constant has a value of 14.

**FW\_ENFORCEMENT\_STATE\_LOCAL\_AUTHORIZED\_APPLICATIONS\_DISALLOWED:** The object is not being enforced because the FW\_PROFILE\_CONFIG\_GLOBAL\_PORTS\_ALLOW\_USER\_PREF\_MERGE configuration option (see section 2.2.37 for more details) from a profile that the object applied to, disallowed its use. This symbolic constant has a value of 15.

**FW\_ENFORCEMENT\_STATE\_LOCAL\_FIREWALL\_RULES\_DISALLOWED:** The object is not being enforced because the FW\_PROFILE\_CONFIG\_ALLOW\_LOCAL\_POLICY\_MERGE configuration option (see section 2.2.37 for more details) from a profile that the object applied to, disallowed its use. This symbolic constant has a value of 16.

**FW\_ENFORCEMENT\_STATE\_LOCAL\_CONSEC\_RULES\_DISALLOWED:** The object is not being enforced because the FW\_PROFILE\_CONFIG\_ALLOW\_LOCAL\_IPSEC\_POLICY\_MERGE configuration option (see section 2.2.37 for more details) from a profile that the object applied to, disallowed its use. This symbolic constant has a value of 17.

**FW\_ENFORCEMENT\_STATE\_MISMATCHED\_PLATFORM:** The object is not being enforced because the platform validity condition does not match the current platform of the host. This symbolic constant has a value of 18.

**FW\_ENFORCEMENT\_STATE\_OPTIMIZED\_OUT:** The object is not being enforced because the firewall and advanced security component determined that the object-implemented functionality is irrelevant (would not change or affect what traffic is allowed or permitted) at the current time. Therefore, the component optimized out the irrelevant functionality and ignored it. This is a pure optimization. This symbolic constant has a value of 19.

**FW\_ENFORCEMENT\_STATE\_LOCAL\_USER\_EMPTY:** The object is not being enforced, because the local user condition of the object contains an SDDL with a SID that is not currently available on the host. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used. This symbolic constant has a value of 20.

**FW\_ENFORCEMENT\_STATE\_TRANSPORT\_MACHINE\_SD\_EMPTY:** The object is not being enforced because the [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) transport mode machine authorization list contains an SDDL with a SID that is not currently available on the host. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used. This symbolic constant has a value of 21.

**FW\_ENFORCEMENT\_STATE\_TRANSPORT\_USER\_SD\_EMPTY:** The object is not being enforced, because the IPsec transport mode user authorization list contains an SDDL with a SID that is not currently available on the host. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used. This symbolic constant has a value of 22.

**FW\_ENFORCEMENT\_STATE\_TUPLE\_RESOLUTION\_EMPTY:** The object is not being enforced, because the trust tuple keywords resolve to an empty set. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used. This symbolic constant has a value of 23.

**FW\_ENFORCEMENT\_STATE\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 24.

### FW\_OBJECT\_METADATA

This structure contains the metadata that is associated with a specific policy object.

1. typedef struct \_tag\_FW\_OBJECT\_METADATA {
2. unsigned \_\_int64 qwFilterContextID;
3. [range(0, 100)] DWORD dwNumEntries;
4. [size\_is(dwNumEntries)] FW\_ENFORCEMENT\_STATE\* pEnforcementStates;
5. } FW\_OBJECT\_METADATA,
6. \*PFW\_OBJECT\_METADATA;

**qwFilterContextID:**  This field is not used across the wires.

**dwNumEntries:**  A field that specifies the number of metadata hints ([FW\_ENFORCEMENT\_STATE](#Section_4767a26e06874c4fad948de700e5cf83)s) that the structure contains.

**pEnforcementStates:**  A pointer to an array of FW\_ENFORCEMENT\_STATE elements. The number of elements is given by **dwNumEntries**.

### FW\_OS\_PLATFORM\_OP

This enumeration describes the operations used in the [FW\_OS\_PLATFORM](#Section_aac2be8cf4ab4b0a995774462137fd8a) structure to determine if an object should be applied to a specified operating system platform.

1. typedef enum
2. {
3. FW\_OS\_PLATFORM\_OP\_EQ = 0,
4. FW\_OS\_PLATFORM\_OP\_GTEQ = 1,
5. FW\_OS\_PLATFORM\_OP\_MAX = 2
6. } FW\_OS\_PLATFORM\_OP;

**FW\_OS\_PLATFORM\_OP\_EQ:** The operating system platform MUST be the same as the one specified. This is satisfied when the following occurs:

If ( ((bPlatform & 0x7) == platform type) && (bMajorVersion == major version) && (bMinorVersion == minor version) ).

Otherwise, the operating system is not equal to the one specified. This symbolic constant has a value of 0.

**FW\_OS\_PLATFORM\_OP\_GTEQ:** The operating system MUST be greater than or equal to the one specified. This is satisfied when any of the following occur:

If (bPlatform & 0x7) > platform type

If (((bPlatform & 0x7) == platform type) && (bMajorVersion > major version))

If (((bPlatform & 0x7) == platform type) && (bMajorVersion == major version) && (bMinorVersion >= minor version))

Otherwise, the operation system is not greater than or equal to the one specified. This symbolic constant has a value of 1.

**FW\_OS\_PLATFORM\_OP\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 2.

### FW\_OS\_PLATFORM

This structure describes a set of operating system platforms. The fields in this data type correspond to the fields of the OSVERSIONINFOEX data type (for more information, see [[MSDN-OSVERSIONINFOEX]](https://go.microsoft.com/fwlink/?LinkId=90057)). There are no constraints on the values allowed for the platform type, major version, or minor version. The set can include values that do not correspond to any existing operating system platform.

1. typedef struct \_tag\_FW\_OS\_PLATFORM {
2. unsigned char bPlatform;
3. unsigned char bMajorVersion;
4. unsigned char bMinorVersion;
5. unsigned char Reserved;
6. } FW\_OS\_PLATFORM,
7. \*PFW\_OS\_PLATFORM;

**bPlatform:**  The three least significant bits identify the platform type. This corresponds to the dwPlatformId field in MSDN. The five most significant bits contain a value from the [FW\_OS\_PLATFORM\_OP](#Section_925113284eed4764a680d2d35214d051) enumeration.

**bMajorVersion:**  Specifies the major version number for the OS. This corresponds to the dwMajorVersion field in MSDN.

**bMinorVersion:**  Specifies the minor version number for the OS. This corresponds to the dwMinorVersion field in MSDN.

**Reserved:**  Not used. Reserved for future use.

### FW\_OS\_PLATFORM\_LIST

This structure contains an array of [FW\_OS\_PLATFORM](#Section_aac2be8cf4ab4b0a995774462137fd8a) elements. The structure describes a set of operating system platforms. This set is the union of the sets identified by each FW\_OS\_PLATFORM element.

1. typedef struct \_tag\_FW\_OS\_PLATFORM\_LIST {
2. [range(0, 1000)] unsigned long dwNumEntries;
3. [size\_is(dwNumEntries)] PFW\_OS\_PLATFORM pPlatforms;
4. } FW\_OS\_PLATFORM\_LIST,
5. \*PFW\_OS\_PLATFORM\_LIST;

**dwNumEntries:**  This field specifies the number of OS platforms that the structure contains.

**pPlatforms:**  A pointer to an array of **dwNumEntries** contiguous FW\_OS\_PLATFORM elements.

### FW\_RULE\_ORIGIN\_TYPE

This enumeration represents where the policy object is stored and from where it originates.

1. typedef enum \_tag\_FW\_RULE\_ORIGIN\_TYPE
2. {
3. FW\_RULE\_ORIGIN\_INVALID = 0,
4. FW\_RULE\_ORIGIN\_LOCAL = 1,
5. FW\_RULE\_ORIGIN\_GP = 2,
6. FW\_RULE\_ORIGIN\_DYNAMIC = 3,
7. FW\_RULE\_ORIGIN\_AUTOGEN = 4,
8. FW\_RULE\_ORIGIN\_HARDCODED = 5,
9. FW\_RULE\_ORIGIN\_MAX = 6
10. } FW\_RULE\_ORIGIN\_TYPE;

**FW\_RULE\_ORIGIN\_INVALID:** On enumeration, this value is invalid, and MUST NOT be used by the server. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. However, the server ignores the fields of this data type on input, and hence it is valid for filling rules. This symbolic constant has a value of 0.

**FW\_RULE\_ORIGIN\_LOCAL:** Specifies that the policy object originates from the local store. This symbolic constant has a value of 1.

**FW\_RULE\_ORIGIN\_GP:** Specifies that the policy object originates from the GP store. This symbolic constant has a value of 2.

**FW\_RULE\_ORIGIN\_DYNAMIC:** Specifies that the policy object originates from the dynamic store. This symbolic constant has a value of 3.

**FW\_RULE\_ORIGIN\_AUTOGEN:** Not used. This symbolic constant has a value of 4.

**FW\_RULE\_ORIGIN\_HARDCODED:** Specifies that the policy object originates from the firewall and advanced security component hard-coded values and is used due to lack of user settings. These values are not configurable and are not addressed in this protocol specification. Specific implementations of firewall and advanced security components can choose what hard-coded values to use when no other user settings are available. The only policy objects in this protocol specification that can have this FW\_RULE\_ORIGIN\_HARDCODED value assigned are authentication sets and cryptographic sets, which are defined in sections [2.2.64](#Section_bbec7ab70a1d49a49c8ea4784bbe6f16) and [2.2.73](#Section_a468fe9e113b4155a63d0db3aac12619), respectively.[<3>](#Appendix_A_3" \o "Product behavior note 3) This symbolic constant has a value of 5.

**FW\_RULE\_ORIGIN\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 6.

### FW\_ENUM\_RULES\_FLAGS

This enumeration defines flag values that can be used in the enumeration methods that are defined in [RRPC\_FWEnumFirewallRules](#Section_469f8466bb5c4af097a8335f648c70d7), [RRPC\_FWEnumConnectionSecurityRules](#Section_f0f85db355104b73a4cb76bd440eaa06), [RRPC\_FWEnumAuthenticationSets](#Section_2dc9c31538e842d69c88927dea828314), and [RRPC\_FWEnumCryptoSets](#Section_63e911eafbca409689a95d84c37daa56).

1. typedef enum \_tag\_FW\_ENUM\_RULES\_FLAGS
2. {
3. FW\_ENUM\_RULES\_FLAG\_NONE = 0x0000,
4. FW\_ENUM\_RULES\_FLAG\_RESOLVE\_NAME = 0x0001,
5. FW\_ENUM\_RULES\_FLAG\_RESOLVE\_DESCRIPTION = 0x0002,
6. FW\_ENUM\_RULES\_FLAG\_RESOLVE\_APPLICATION = 0x0004,
7. FW\_ENUM\_RULES\_FLAG\_RESOLVE\_KEYWORD = 0x0008,
8. FW\_ENUM\_RULES\_FLAG\_RESOLVE\_GPO\_NAME = 0x0010,
9. FW\_ENUM\_RULES\_FLAG\_EFFECTIVE = 0x0020,
10. FW\_ENUM\_RULES\_FLAG\_INCLUDE\_METADATA = 0x0040,
11. FW\_ENUM\_RULES\_FLAG\_MAX = 0x0080
12. } FW\_ENUM\_RULES\_FLAGS;

**FW\_ENUM\_RULES\_FLAG\_NONE:** This value signifies that no specific flag is used. It is defined for [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code to add readability, instead of using the number 0. This symbolic constant has a value 0x0000.

**FW\_ENUM\_RULES\_FLAG\_RESOLVE\_NAME:** Resolves rule description strings to user-friendly, localizable strings if they are in the following format: @file.dll,-<resID>. resID refers to the resource ID in the indirect string. Please see [[MSDN-SHLoadIndirectString]](https://go.microsoft.com/fwlink/?LinkId=210820) for further documentation on the string format. This symbolic constant has a value 0x0001.

**FW\_ENUM\_RULES\_FLAG\_RESOLVE\_DESCRIPTION:** Resolves rule description strings to user-friendly, localizable strings if they are in the following format: @file.dll,-<resID>. resID refers to the resource ID in the indirect string. Please see [MSDN-SHLoadIndirectString] for further documentation on the string format. This symbolic constant has a value 0x0002.

**FW\_ENUM\_RULES\_FLAG\_RESOLVE\_APPLICATION:** If this flag is set, the server MUST inspect the **wszLocalApplication** field of each **FW\_RULE** structure and replace all environment variables in the string with their corresponding values. See [[MSDN-ExpandEnvironmentStrings]](https://go.microsoft.com/fwlink/?LinkId=211083) for more details about environment-variable strings. This symbolic constant has a value 0x0004.

**FW\_ENUM\_RULES\_FLAG\_RESOLVE\_KEYWORD:** Resolves keywords in addresses and ports to the actual addresses and ports (dynamic store only). This symbolic constant has a value 0x0008.

**FW\_ENUM\_RULES\_FLAG\_RESOLVE\_GPO\_NAME:** Resolves the [**GPO**](#gt_dec32233-8776-4151-91a0-8624a2b9abb0) name for the GP\_RSOP rules. This symbolic constant has a value 0x0010.

**FW\_ENUM\_RULES\_FLAG\_EFFECTIVE:** If this flag is set, the server MUST only return objects where at least one FW\_ENFORCEMENT\_STATE entry in the object's metadata is equal to FW\_ENFORCEMENT\_STATE\_FULL. This flag is available for the dynamic store only. This symbolic constant has a value 0x0020.

**FW\_ENUM\_RULES\_FLAG\_INCLUDE\_METADATA:** Includes the metadata object information, represented by the **FW\_OBJECT\_METADATA** structure, in the enumerated objects. This symbolic constant has a value 0x0040.

**FW\_ENUM\_RULES\_FLAG\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value 0x0080.

### FW\_RULE\_ACTION

This enumeration describes the possible actions on firewall rules.

1. typedef enum \_tag\_FW\_RULE\_ACTION
2. {
3. FW\_RULE\_ACTION\_INVALID = 0,
4. FW\_RULE\_ACTION\_ALLOW\_BYPASS = 1,
5. FW\_RULE\_ACTION\_BLOCK = 2,
6. FW\_RULE\_ACTION\_ALLOW = 3,
7. FW\_RULE\_ACTION\_MAX = 4
8. } FW\_RULE\_ACTION;

**FW\_RULE\_ACTION\_INVALID:** This value is invalid and MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 0.

**FW\_RULE\_ACTION\_ALLOW\_BYPASS:** Rules with this action allow traffic but are applicable only to rules that at least specify the **FW\_RULE\_FLAGS\_AUTHENTICATE** flag. This symbolic constant has a value of 1.

**FW\_RULE\_ACTION\_BLOCK:** Rules with this action block traffic. This symbolic constant has a value of 2.

**FW\_RULE\_ACTION\_ALLOW:** Rules with this action allow traffic. This symbolic constant has a value of 3.

**FW\_RULE\_ACTION\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 4.

If conflicting rules match the same network traffic, the actions determine the order of precedence. Allow-bypass rules take precedence over block rules, and block rules take precedence over allow rules.

### FW\_RULE\_FLAGS

This enumeration represents flags that can be specified in firewall rules of section [2.2.36](#Section_8c008258166d46d49090f2ffaa01be4b).

1. typedef enum \_tag\_FW\_RULE\_FLAGS
2. {
3. FW\_RULE\_FLAGS\_NONE = 0x0000,
4. FW\_RULE\_FLAGS\_ACTIVE = 0x0001,
5. FW\_RULE\_FLAGS\_AUTHENTICATE = 0x0002,
6. FW\_RULE\_FLAGS\_AUTHENTICATE\_WITH\_ENCRYPTION = 0x0004,
7. FW\_RULE\_FLAGS\_ROUTEABLE\_ADDRS\_TRAVERSE = 0x0008,
8. FW\_RULE\_FLAGS\_LOOSE\_SOURCE\_MAPPED = 0x0010,
9. FW\_RULE\_FLAGS\_MAX\_V2\_1 = 0x0020,
10. FW\_RULE\_FLAGS\_AUTH\_WITH\_NO\_ENCAPSULATION = 0x0020,
11. FW\_RULE\_FLAGS\_MAX\_V2\_9 = 0x0040,
12. FW\_RULE\_FLAGS\_AUTH\_WITH\_ENC\_NEGOTIATE = 0x0040,
13. FW\_RULE\_FLAGS\_ROUTEABLE\_ADDRS\_TRAVERSE\_DEFER\_APP = 0x0080,
14. FW\_RULE\_FLAGS\_ROUTEABLE\_ADDRS\_TRAVERSE\_DEFER\_USER = 0x0100,
15. FW\_RULE\_FLAGS\_AUTHENTICATE\_BYPASS\_OUTBOUND = 0x0200,
16. FW\_RULE\_FLAGS\_MAX\_V2\_10 = 0x0400,
17. FW\_RULE\_FLAGS\_ALLOW\_PROFILE\_CROSSING = 0x0400,
18. FW\_RULE\_FLAGS\_LOCAL\_ONLY\_MAPPED = 0x0800,
19. FW\_RULE\_FLAGS\_MAX\_V2\_20 = 0x1000,
20. FW\_RULE\_FLAGS\_LUA\_CONDITIONAL\_ACE = 0x1000,
21. FW\_RULE\_FLAGS\_BIND\_TO\_INTERFACE = 0x2000,
22. FW\_RULE\_FLAGS\_MAX = 0x4000,
23. } FW\_RULE\_FLAGS;

**FW\_RULE\_FLAGS\_NONE:** This value means that none of the following flags are set. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code.

**FW\_RULE\_FLAGS\_ACTIVE:** The rule is enabled if this flag is set; otherwise, it is disabled.

**FW\_RULE\_FLAGS\_AUTHENTICATE:** This flag MUST be set only on rules that have the allow actions. If set, traffic that matches the rule is allowed only if it has been authenticated by [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb); otherwise, traffic is blocked.

**FW\_RULE\_FLAGS\_AUTHENTICATE\_WITH\_ENCRYPTION:** This flag is similar to the FW\_RULE\_FLAGS\_AUTHENTICATE flag; however, traffic MUST also be encrypted.

**FW\_RULE\_FLAGS\_ROUTEABLE\_ADDRS\_TRAVERSE:** This flag MUST be set only on inbound rules. This flag allows the matching traffic to traverse a NAT edge device and be allowed in the host computer.

**FW\_RULE\_FLAGS\_LOOSE\_SOURCE\_MAPPED:** This flag allows responses from a remote IP address that is different from the one to which the outbound matched traffic originally went.

**FW\_RULE\_FLAGS\_AUTH\_WITH\_NO\_ENCAPSULATION:** This flag MUST be set only on rules that have the FW\_RULE\_FLAGS\_AUTHENTICATE flag set. If set, traffic that matches the rule is allowed if [**IKE**](#gt_294fef97-5790-4d41-971e-dd255b783e68) or [**AuthIP**](#gt_3791f3e1-cf2f-4605-9fcc-54f526f036cf) authentication was successful; however, this flag does not necessarily require that traffic be protected by IPsec encapsulations. For schema versions 0x0200 and 0x0201, this value is invalid and MUST NOT be used.

**FW\_RULE\_FLAGS\_AUTH\_WITH\_ENC\_NEGOTIATE:** This flag MUST be set only on inbound rules that have the FW\_RULE\_FLAGS\_AUTHENTICATE\_WITH\_ENCRYPTION flag set. If set and if the first packet that arrives is unencrypted but authenticated by IPsec, the packet is allowed, and an IKE or AuthIP negotiation is started to negotiate encryption settings and encrypt subsequent packets. [[MS-AIPS]](%5bMS-AIPS%5d.pdf#Section_eee3de6438474451978e9513ff187d30) section 3.2.4 specifies negotiation initiation behavior for hosts that support both IKE and AuthIP negotiation. If the negotiation fails, the connection is dropped. For schema versions 0x0200 and 0x0201, this value is invalid and MUST NOT be used.

**FW\_RULE\_FLAGS\_ROUTEABLE\_ADDRS\_TRAVERSE\_DEFER\_APP:** This flag MUST be set only on inbound rules. This flag allows the matching traffic to traverse a NAT edge device and be allowed in the host computer, if and only if a matching **PortInUse** object is found in the **PortsInUse** collection with **NATTraversalRequested** set to true (see section [3.1.1](#Section_43507d538955416db913dfb27dc76b17)). For schema versions 0x0200 and 0x0201, this value is invalid and MUST NOT be used.

**FW\_RULE\_FLAGS\_ROUTEABLE\_ADDRS\_TRAVERSE\_DEFER\_USER:** This flag MUST be set only on inbound rules. Whenever an application tries to listen for traffic that matches this rule, the operating system asks the administrator of the host whether it should allow this traffic to traverse the NAT. For schema versions 0x0200 and 0x0201, this value is invalid and MUST NOT be used.

**FW\_RULE\_FLAGS\_AUTHENTICATE\_BYPASS\_OUTBOUND:** This flag MUST be set only on outbound rules that have an allow action with either the FW\_RULE\_FLAGS\_AUTHENTICATE or the FW\_RULE\_FLAGS\_AUTHENTICATE\_WITH\_ENCRYPTION flag set. If set, this rule is evaluated before block rules, making it equivalent to a rule with an FW\_RULE\_ACTION\_ALLOW\_BYPASS, but for outbound. For schema versions 0x0200 and 0x0201, this value is invalid and MUST NOT be used.

**FW\_RULE\_FLAGS\_ALLOW\_PROFILE\_CROSSING:** This flag allows responses from a network with a different profile type than the network to which the outbound traffic was originally sent. This flag MUST be ignored on rules with an action of FW\_RULE\_ACTION\_BLOCK. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used.

**FW\_RULE\_FLAGS\_LOCAL\_ONLY\_MAPPED:** If this flag is set on a rule, the remote address and remote port conditions are ignored when determining whether a network traffic flow matches the rule. This flag MUST be ignored on rules with an action of FW\_RULE\_ACTION\_BLOCK. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used.

**FW\_RULE\_FLAGS\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 0x4000.

**FW\_RULE\_FLAGS\_MAX\_V2\_1:** This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x0201 and earlier. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 0x0020.

**FW\_RULE\_FLAGS\_MAX\_V2\_9:** This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x0209 and earlier. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 0x0040.

**FW\_RULE\_FLAGS\_MAX\_V2\_10:** This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x020A and earlier. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 0x0400.

**FW\_RULE\_FLAGS\_MAX\_V2\_20:** This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x0214 and earlier. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 0x1000.

**FW\_RULE\_FLAGS\_LUA\_CONDITIONAL\_ACE:** This flag MUST be set if and only if the **wszLocalUserAuthorizationList** field of the [FW\_RULE2\_24](#Section_4f5bc5fd4a3c46a598dc98b31b73aa89) structure (section 2.2.103) is to include conditional [**ACEs**](#gt_b581857f-39aa-4979-876b-daba67a40f15). For schema versions 0x0200, 0x0201, 0x020A, 0x0214, and 0x0216, this value is invalid and MUST NOT be used.

**FW\_RULE\_FLAGS\_BIND\_TO\_INTERFACE**: This flag is not used.

### FW\_RULE2\_0

This structure represents a firewall rule that is used by the 2.0 binary version servers and clients (see sections [1.7](#Section_6b14b8f991ab4f09873f9e2334196cdc) and [2.2.41](#Section_faf4ffbe1d5140adae902230f2c0b6a9)). The fields of this structure are identical to the [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b) structure and its meanings are covered in section 2.2.36.

1. typedef struct \_tag\_FW\_RULE2\_0 {
2. struct \_tag\_FW\_RULE2\_0\* pNext;
3. unsigned short wSchemaVersion;
4. [string, range(1, 10001), ref] wchar\_t\* wszRuleId;
5. [string, range(1, 10001)] wchar\_t\* wszName;
6. [string, range(1, 10001)] wchar\_t\* wszDescription;
7. unsigned long dwProfiles;
8. [range(FW\_DIR\_INVALID, FW\_DIR\_OUT)]
9. FW\_DIRECTION Direction;
10. [range(0, 256)] unsigned short wIpProtocol;
11. [switch\_type(unsigned short), switch\_is(wIpProtocol)]
12. union {
13. [case(6,17)]
14. struct {
15. FW\_PORTS LocalPorts;
16. FW\_PORTS RemotePorts;
17. };
18. [case(1)]
19. FW\_ICMP\_TYPE\_CODE\_LIST V4TypeCodeList;
20. [case(58)]
21. FW\_ICMP\_TYPE\_CODE\_LIST V6TypeCodeList;
22. [default]  ;
23. };
24. FW\_ADDRESSES LocalAddresses;
25. FW\_ADDRESSES RemoteAddresses;
26. FW\_INTERFACE\_LUIDS LocalInterfaceIds;
27. unsigned long dwLocalInterfaceTypes;
28. [string, range(1, 10001)] wchar\_t\* wszLocalApplication;
29. [string, range(1, 10001)] wchar\_t\* wszLocalService;
30. [range(FW\_RULE\_ACTION\_INVALID, FW\_RULE\_ACTION\_MAX)]
31. FW\_RULE\_ACTION Action;
32. unsigned short wFlags;
33. [string, range(1, 10001)] wchar\_t\* wszRemoteMachineAuthorizationList;
34. [string, range(1, 10001)] wchar\_t\* wszRemoteUserAuthorizationList;
35. [string, range(1, 10001)] wchar\_t\* wszEmbeddedContext;
36. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
37. FW\_RULE\_STATUS Status;
38. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX)]
39. FW\_RULE\_ORIGIN\_TYPE Origin;
40. [string, range(1, 10001)] wchar\_t\* wszGPOName;
41. unsigned long Reserved;
42. } FW\_RULE2\_0,
43. \*PFW\_RULE2\_0;

### FW\_RULE

This structure is used to represent a firewall rule.

1. typedef struct \_tag\_FW\_RULE {
2. struct \_tag\_FW\_RULE\* pNext;
3. unsigned short wSchemaVersion;
4. [string, range(1, 512), ref] wchar\_t\* wszRuleId;
5. [string, range(1, 10001)] wchar\_t\* wszName;
6. [string, range(1, 10001)] wchar\_t\* wszDescription;
7. unsigned long dwProfiles;
8. [range(FW\_DIR\_INVALID, FW\_DIR\_OUT)]
9. FW\_DIRECTION Direction;
10. [range(0, 256)] unsigned short wIpProtocol;
11. [switch\_type(unsigned short), switch\_is(wIpProtocol)]
12. union {
13. [case(6,17)]
14. struct {
15. FW\_PORTS LocalPorts;
16. FW\_PORTS RemotePorts;
17. };
18. [case(1)]
19. FW\_ICMP\_TYPE\_CODE\_LIST V4TypeCodeList;
20. [case(58)]
21. FW\_ICMP\_TYPE\_CODE\_LIST V6TypeCodeList;
22. [default] ;
23. };
24. FW\_ADDRESSES LocalAddresses;
25. FW\_ADDRESSES RemoteAddresses;
26. FW\_INTERFACE\_LUIDS LocalInterfaceIds;
27. unsigned long dwLocalInterfaceTypes;
28. [string, range(1, 10001)] wchar\_t\* wszLocalApplication;
29. [string, range(1, 10001)] wchar\_t\* wszLocalService;
30. [range(FW\_RULE\_ACTION\_INVALID, FW\_RULE\_ACTION\_MAX)]
31. FW\_RULE\_ACTION Action;
32. unsigned short wFlags;
33. [string, range(1, 10001)] wchar\_t\* wszRemoteMachineAuthorizationList;
34. [string, range(1, 10001)] wchar\_t\* wszRemoteUserAuthorizationList;
35. [string, range(1, 10001)] wchar\_t\* wszEmbeddedContext;
36. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
37. FW\_RULE\_STATUS Status;
38. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX)]
39. FW\_RULE\_ORIGIN\_TYPE Origin;
40. [string, range(1, 10001)] wchar\_t\* wszGPOName;
41. unsigned long Reserved;
42. [size\_is((Reserved & FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA) ? 1 : 0)]
43. PFW\_OBJECT\_METADATA pMetaData;
44. [string, range(1, 10001)] WCHAR\* wszLocalUserAuthorizationList;
45. [string, range(1, 10001)] WCHAR\* wszPackageId;
46. [string, range(1, 10001)] WCHAR\* wszLocalUserOwner;
47. unsigned long dwTrustTupleKeywords;
48. FW\_NETWORK\_NAMES OnNetworkNames;
49. [string, range(1, 10001)] WCHAR\* wszSecurityRealmId;
50. unsigned short wFlags2;
51. FW\_NETWORK\_NAMES RemoteOutServerNames;
52. [string, range(1,10001)] WCHAR\* wszFqbn;
53. unsigned long compartmentId;
54. } FW\_RULE,
55. \*PFW\_RULE;

**pNext:**  A pointer to the next FW\_RULE in the list.

**wSchemaVersion:**  Specifies the version of the rule.

**wszRuleId:**  A pointer to a [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string that uniquely identifies the rule.

**wszName:**  A pointer to a Unicode string that provides a friendly name for the rule.

**wszDescription:**  A pointer to a Unicode string that provides a friendly description for the rule.

**dwProfiles:**  A bitmask of the [FW\_PROFILE\_TYPE](#Section_7704e238174d4a5eb8095f3787dd8acc) flags. It is a condition that matches traffic on the specified profiles.

**Direction:**  Specifies the direction of the traffic that the rule matches.

**wIpProtocol:**  A condition that specifies the protocol of the traffic that the rule matches. If the value is within the range 0 to 255, the value describes a protocol in IETF IANA numbers (for more information, see [[IANA-PROTO-NUM]](https://go.microsoft.com/fwlink/?LinkId=89889)). If the value is 256, the rule matches any protocol.

**LocalPorts:**  A condition that specifies the local host ports of the [**TCP**](#gt_b08d36f6-b5c6-4ce4-8d2d-6f2ab75ea4cb) or UDP traffic that the rule matches.

**RemotePorts:**  A condition that specifies the remote host ports of the TCP or UDP traffic that the rule matches.

**V4TypeCodeList:**  A condition that specifies the list of ICMP types of the traffic that the rule matches. This field applies only when **wIpProtocol** specifies ICMP v4.

**V6TypeCodeList:**  A condition that specifies the list of ICMP types of the traffic that the rule matches. This field applies only when **wIpProtocol** specifies ICMP v6.

**LocalAddresses:**  A condition that specifies the addresses of the local host of the traffic that the rule matches. An empty **LocalAddresses** structure means that this condition is not applied.

**RemoteAddresses:**  A condition that specifies the addresses of the remote host of the traffic that the rule matches. An empty **RemoteAddresses** structure means that this condition is not applied.

**LocalInterfaceIds:**  A condition that specifies the list of specific network interfaces used by the traffic that the rule matches. A **LocalInterfaceIds** field with no interface [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) specified means that the rule applies to all interfaces; that is, the condition is not applied.

**dwLocalInterfaceTypes:**  A bitmask of [FW\_INTERFACE\_TYPE](#Section_d067c0284ea943199ddf0fb51c191b35). It is a condition that restricts the interface types that are used by the traffic that the rule matches. 0x00000000 means that the condition matches all interface types.

**wszLocalApplication:**  A pointer to a Unicode string. It is a condition that specifies a file path name to the executable that uses the traffic that the rule matches. A null in this field means that the rule applies to all processes in the host.

**wszLocalService:**  A pointer to a Unicode string. It is a condition that specifies the service name of the service that uses the traffic that the rule matches. An L"\*" string in this field means that the rule applies to all services in the system. A null in this field means that the rule applies to all processes.

**Action:**  The action that the rule will take for the traffic matches.

**wFlags:**  Bit flags from [FW\_RULE\_FLAGS](#Section_fc67ea0419f04ccd8912abe467e7c11a).

**wszRemoteMachineAuthorizationList:**  A pointer to a Unicode string. A condition that specifies the remote machines sending or receiving the traffic that the rule matches. The string is in SDDL format ([[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.5.1).

**wszRemoteUserAuthorizationList:**  A pointer to a Unicode string. A condition that specifies the remote users accepting or receiving the traffic that the rule matches. The string is in SDDL format ([MS-DTYP] section 2.5.1).

**wszEmbeddedContext:**  A pointer to a Unicode string. It specifies a group name for this rule. Other components in the system use this string to enable or disable groups of rules by verifying that they all have the same group name.

**PlatformValidityList:**  A condition in a rule that determines whether or not the rule is enforced by the local computer based on the local computer's platform information. The rule is enforced only if the local computer's operating system platform is an element of the set described by **PlatformValidityList**.[<4>](#Appendix_A_4" \o "Product behavior note 4)

**Status:**  The status code of the rule, as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out when the structure is returned as output. On input, this field MUST be set to FW\_RULE\_STATUS\_OK.

**Origin:**  The rule origin, as specified in the [FW\_RULE\_ORIGIN\_TYPE](#Section_9d295321d75c41c0ab0d7a78df40f77c) enumeration. It MUST be filled on enumerated rules and ignored on input.

**wszGPOName:**  A pointer to a Unicode string containing the displayName of the [**GPO**](#gt_dec32233-8776-4151-91a0-8624a2b9abb0) containing this object. When adding a new object, this field is not used. The client SHOULD set the value to NULL, and the server MUST ignore the value. When enumerating an existing object, if the client does not set the FW\_ENUM\_RULES\_FLAG\_RESOLVE\_GPO\_NAME flag, the server MUST set the value to NULL. Otherwise, the server MUST set the value to the displayName of the GPO containing the object or NULL if the object is not contained within a GPO. For details about how the server initializes an object from a GPO, see section [3.1.3](#Section_e8924ac5aa4a41d1bf654f46b3d399aa). For details about how the displayName of a GPO is stored, see [[MS-GPOL]](%5bMS-GPOL%5d.pdf#Section_62d1292462524052996f161d2b9019f4) section 2.3.

**Reserved:**  Not used other than to instruct [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331), using the FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA flag, that a pointer to an FW\_OBJECT\_METADATA structure is present. It has no semantic meaning to the object itself.

**pMetaData:**  A pointer to an FW\_OBJECT\_METADATA structure that contains specific metadata about the current state of the firewall rule.

**wszLocalUserAuthorizationList:**  A pointer to a Unicode string in SDDL format ([MS-DTYP] section 2.5.1). It is a condition that specifies the local users accepting or receiving the traffic that the rule matches.

**wszPackageId:**  A pointer to a Unicode string in [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) string format ([MS-DTYP] section 2.4.2.1). It is a condition that specifies the application SID of the process that uses the traffic that the rule matches. A null in this field means that the rule applies to all processes in the host.

**wszLocalUserOwner:**  A pointer to a Unicode string in SID string format. The SID specifies the security principal that owns the rule.

**dwTrustTupleKeywords:**  A bitmask of the FW\_TRUST\_TUPLE\_KEYWORD flags. It is a condition that matches traffic associated with the specified trust tuples.

**OnNetworkNames**: Specifies the networks, identified by name, in which the rule must be enforced.

**wszSecurityRealmId**: A pointer to a Unicode string in SID string format. The SID specifies the Security Realm ID, which identifies a security realm that this firewall rule is associated with. Any application that matches this rule will be subject to the IPsec polices for this security realm.

**wFlags2**: Bit flags from FW\_RULE\_FLAGS2 (section 2.2.102).

**RemoteOutServerNames**: This value is not used over the wire.

**wszFqbn**: A string that is formatted as an FQBN; also see [[MSDN-FQBN]](https://go.microsoft.com/fwlink/?linkid=839018).

**compartmentId**: The ID of the compartment or Windows Server Container.

The following are semantic checks that firewall rules MUST pass:

* The **wSchemaVersion** field MUST NOT be less than 0x000100.
* The **wSchemaVersion** field SHOULD NOT be less than 0x000200.[<5>](#Appendix_A_5" \o "Product behavior note 5)
* The **wszRuleId** field MUST NOT contain the pipe (|) character, MUST NOT be NULL, MUST be a string of at least 1 character, and MUST NOT be greater or equal to 512 characters.[<6>](#Appendix_A_6" \o "Product behavior note 6)
* The **wszName** field string MUST meet the following criteria:
  + MUST contain 1 or more characters.
  + MUST contain fewer than 10,000 characters.
  + MUST NOT be NULL.
  + MUST NOT contain the pipe (|) character.
  + MUST NOT equal the case-insensitive string "ALL".
* If the **wszDescription** field string is not NULL, it MUST contain at least 1 character, MUST NOT be greater than or equal to 10,000 characters, and MUST NOT contain the pipe (|) character.
* If the **wszLocalApplication** field string is not NULL, it MUST be at least 1 character, MUST NOT be greater than or equal to MAX\_PATH (260) characters, and MUST NOT contain the following characters: /,\*,?,",<,>,|.
* If the **wszLocalService** field string is not NULL, it MUST contain at least 1 character, MUST NOT be greater than or equal to MAX\_PATH characters, and MUST NOT contain the following characters: /,\,|.
* If the **wszEmbeddedContext** field string is not NULL, it MUST contain at least 1 character, MUST NOT be greater than or equal to 10,000 characters, and MUST NOT contain the pipe (|) character.
* The **Direction** field MUST NOT contain invalid [FW\_DIRECTION](#Section_34806362c31f4531ad8c5a43d5870223) values.
* The **dwProfiles** field MUST NOT contain invalid values and, if it is not equal to the FW\_PROFILE\_TYPE\_ALL profile type, it MUST NOT contain unknown profiles.
* The **wIpProtocol** field MUST NOT be greater than 256.
* If the **wPortKeywords** field of **LocalPorts** is FW\_PORT\_KEYWORD\_DYNAMIC\_RPC\_PORTS or FW\_PORT\_KEYWORD\_RPC\_EP, the **wIpProtocol** field MUST be 6, and **Direction** MUST be FW\_DIRECTION\_IN.
* If the **wPortKeywords** field of **LocalPorts** is FW\_PORT\_KEYWORD\_TEREDO\_PORT, the **wIpProtocol** field MUST be 17, and **Direction** MUST be FW\_DIRECTION\_IN.
* The **wPortKeywords** field of **LocalPorts** MUST be 0 if the **Direction** is FW\_DIRECTION\_OUT.
* If the **wIpProtocol** field is 6 or 17, the **wPortKeywords** field of **RemotePorts** MUST be 0.
* If the **wIpProtocol** field is not 1, 6, 17, or 58, the **LocalPorts**, **RemotePorts**, **V4TypeCodeList**, and **V6TypeCodeList** field MUST be empty.
* The **dwV4AddressKeywords** and **dwV6AddressKeywords** fields of **LocalAddresses** MUST be 0.
* **dwLocalInterfaceTypes** MUST NOT be greater than or equal to FW\_INTERFACE\_TYPE\_MAX.
* **Action** MUST be a valid action from the [FW\_RULE\_ACTION](#Section_702e3c23c9d843db8380b4c670dd7f7d) enumeration.
* **wFlags** MUST NOT be greater than FW\_RULE\_FLAGS\_MAX.
* If **Direction** is FW\_DIR\_OUT, **wFlags** MUST NOT contain a FW\_RULE\_FLAGS\_ROUTEABLE\_ADDRS\_TRAVERSE.
* If **Direction** is FW\_DIR\_IN or **wIpProtocol** is 6 or **wFlags** contains FW\_RULE\_FLAGS\_AUTHENTICATE or FW\_RULE\_FLAGS\_AUTHENTICATE\_WITH\_ENCRYPTION, **wFlags** MUST NOT contain FW\_RULE\_FLAGS\_LOOSE\_SOURCE\_MAPPED.
* The **wFlags** field MUST NOT contain both FW\_RULE\_FLAGS\_AUTHENTICATE and FW\_RULE\_FLAGS\_AUTHENTICATE\_WITH\_ENCRYPTION.
* If **wFlags** contains either FW\_RULE\_FLAGS\_AUTHENTICATE or FW\_RULE\_FLAGS\_AUTHENTICATE\_WITH\_ENCRYPTION, **Action** MUST NOT be FW\_RULE\_ACTION\_BLOCK.
* If **Action** is FW\_RULE\_ACTION\_ALLOW\_BYPASS, **Direction** MUST be FW\_DIR\_IN, **wFlags** MUST contain either FW\_RULE\_FLAGS\_AUTHENTICATE or FW\_RULE\_FLAGS\_AUTHENTICATE\_WITH\_ENCRYPTION, and **wszRemoteMachineAuthorizationList** MUST NOT be NULL.
* If **wszRemoteMachineAuthorizationList** is not NULL, it MUST be at least 1 character, MUST NOT be greater than or equal to 10,000 characters, MUST NOT contain the pipe (|) character, MUST NOT be an empty string (""), MUST be a valid security descriptor ([MS-DTYP] section 2.4.6), MUST have a non-Null [**ACL**](#gt_9f92aa05-dd0a-45f2-88d6-89f1fb654395), MUST have only either Allow or Deny [**ACEs**](#gt_b581857f-39aa-4979-876b-daba67a40f15), and each ACE MUST have a Filter match access right.
* If **wszRemoteUserAuthorizationList** is not NULL, it MUST be at least 1 character, MUST NOT be greater than or equal to 10,000 characters, MUST NOT contain the pipe (|) character, MUST NOT be an empty string (""), MUST be a valid security descriptor ([MS-DTYP] section 2.4.6), MUST have a non-NULL ACL, MUST only have either Allow or Deny ACEs, and each ACE MUST have a Filter match access right.
* If **wszRemoteMachineAuthorizationList** is not NULL or **wszRemoteUserAuthorizationList** is not NULL, either the FW\_RULE\_FLAGS\_AUTHENTICATE flag or the FW\_RULE\_FLAGS\_AUTHENTICATE\_WITH\_ENCRYPT flag MUST be set on the **wFlags** field.
* If the **Direction** field is FW\_DIR\_OUT, the **wszRemoteMachineAuthorizationList** field MUST be NULL.
* If **wszLocalUserAuthorizationList** is not NULL, it MUST be at least 1 character, MUST NOT be greater than or equal to 10,000 characters, MUST NOT contain the pipe ("|") character unless it contains a conditional ACE and the **wFlags** field has the FW\_RULE\_FLAGS\_LUA\_CONDITIONAL\_ACE set (section 2.2.34), MUST NOT be an empty string (""), MUST be a valid security descriptor ([MS-DTYP] section 2.4.6), MUST have a non-NULL ACL, MUST only have either Allow or Deny ACEs if the FW\_RULE\_FLAGS\_LUA\_CONDITIONAL\_ACE is not set, or can include conditional ACEs if FW\_RULE\_FLAGS\_LUA\_CONDITIONAL\_ACE is set, and each ACE MUST have a Filter match access right.

### FW\_PROFILE\_CONFIG

This enumeration identifies each of the per-profile configuration options supported by this protocol. Each configuration option has a merge law that is used to determine how to merge the values of these options across stores.

1. typedef enum \_tag\_FW\_PROFILE\_CONFIG
2. {
3. FW\_PROFILE\_CONFIG\_INVALID = 0,
4. FW\_PROFILE\_CONFIG\_ENABLE\_FW = 1,
5. FW\_PROFILE\_CONFIG\_DISABLE\_STEALTH\_MODE = 2,
6. FW\_PROFILE\_CONFIG\_SHIELDED = 3,
7. FW\_PROFILE\_CONFIG\_DISABLE\_UNICAST\_RESPONSES\_TO\_MULTICAST\_BROADCAST = 4,
8. FW\_PROFILE\_CONFIG\_LOG\_DROPPED\_PACKETS = 5,
9. FW\_PROFILE\_CONFIG\_LOG\_SUCCESS\_CONNECTIONS = 6,
10. FW\_PROFILE\_CONFIG\_LOG\_IGNORED\_RULES = 7,
11. FW\_PROFILE\_CONFIG\_LOG\_MAX\_FILE\_SIZE = 8,
12. FW\_PROFILE\_CONFIG\_LOG\_FILE\_PATH = 9,
13. FW\_PROFILE\_CONFIG\_DISABLE\_INBOUND\_NOTIFICATIONS = 10,
14. FW\_PROFILE\_CONFIG\_AUTH\_APPS\_ALLOW\_USER\_PREF\_MERGE = 11,
15. FW\_PROFILE\_CONFIG\_GLOBAL\_PORTS\_ALLOW\_USER\_PREF\_MERGE = 12,
16. FW\_PROFILE\_CONFIG\_ALLOW\_LOCAL\_POLICY\_MERGE = 13,
17. FW\_PROFILE\_CONFIG\_ALLOW\_LOCAL\_IPSEC\_POLICY\_MERGE = 14,
18. FW\_PROFILE\_CONFIG\_DISABLED\_INTERFACES = 15,
19. FW\_PROFILE\_CONFIG\_DEFAULT\_OUTBOUND\_ACTION = 16,
20. FW\_PROFILE\_CONFIG\_DEFAULT\_INBOUND\_ACTION = 17,
21. FW\_PROFILE\_CONFIG\_DISABLE\_STEALTH\_MODE\_IPSEC\_SECURED\_PACKET\_EXEMPTION = 18,
22. FW\_PROFILE\_CONFIG\_MAX = 19
23. } FW\_PROFILE\_CONFIG;

**FW\_PROFILE\_CONFIG\_INVALID:** This value is invalid and MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 0.

**FW\_PROFILE\_CONFIG\_ENABLE\_FW:** This value is an on/off switch for the firewall and advanced security enforcement. It is a **DWORD** type value; 0x00000000 is off; 0x00000001 is on. If this value is off, the server MUST NOT block any network traffic, regardless of other policy settings. The merge law for this option is to let the value of the **GroupPolicyRSoPStore** win if it is configured; otherwise, the local store value is used. This symbolic constant has a value of 1.

**FW\_PROFILE\_CONFIG\_DISABLE\_STEALTH\_MODE:** This value is a **DWORD** used as an on/off switch. When this option is off, the server operates in [**stealth mode**](#gt_60f8407f-3974-42e5-b542-9156dcd18b82). The firewall rules used to enforce stealth mode are implementation-specific.[<7>](#Appendix_A_7" \o "Product behavior note 7) The merge law for this option is to let the value of the **GroupPolicyRSoPStore** win if it is configured; otherwise, the local store value is used. This symbolic constant has a value of 2.

**FW\_PROFILE\_CONFIG\_SHIELDED:** This value is a **DWORD** used as an on/off switch. If this value is on and FW\_PROFILE\_CONFIG\_ENABLE\_FW is on, the server MUST block all incoming traffic regardless of other policy settings. The merge law for this option is to let "on" values win. This symbolic constant has a value of 3.

**FW\_PROFILE\_CONFIG\_DISABLE\_UNICAST\_RESPONSES\_TO\_MULTICAST\_BROADCAST:** This value is a **DWORD** used as an on/off switch. If it is on, unicast responses to multicast broadcast traffic is blocked. The merge law for this option is to let the value of the **GroupPolicyRSoPStore** win if it is configured; otherwise, the local store value is used. This symbolic constant has a value of 4.

**FW\_PROFILE\_CONFIG\_LOG\_DROPPED\_PACKETS:** This value is a **DWORD** used as an on/off switch. If this value is on, the firewall logs all the dropped packets. The merge law for this option is to let "on" values win. This symbolic constant has a value of 5.

**FW\_PROFILE\_CONFIG\_LOG\_SUCCESS\_CONNECTIONS:** This value is a **DWORD** used as an on/off switch. If this value is on, the firewall logs all successful inbound connections. The merge law for this option is to let "on" values win. This symbolic constant has a value of 6.

**FW\_PROFILE\_CONFIG\_LOG\_IGNORED\_RULES:** This value is a **DWORD** used as an on/off switch. The server MAY use this value in an implementation-specific way to control logging of events if a rule is not enforced for any reason. The merge law for this option is to let "on" values win. This symbolic constant has a value of 7.[<8>](#Appendix_A_8" \o "Product behavior note 8)

**FW\_PROFILE\_CONFIG\_LOG\_MAX\_FILE\_SIZE:** This value is a **DWORD** and specifies the size, in kilobytes, of the log where dropped packets and successful connections are logged. The merge law for this option is to let the value of the **GroupPolicyRSoPStore** win if it is configured; otherwise, the local store value is used. This symbolic constant has a value of 8.

**FW\_PROFILE\_CONFIG\_LOG\_FILE\_PATH:** This configuration value is a string that represents a file path to the log for when the firewall logs dropped packets and successful connections. The merge law for this option is to let the value of the **GroupPolicyRSoPStore** win if it is configured; otherwise, the local store value is used. This symbolic constant has a value of 9.

**FW\_PROFILE\_CONFIG\_DISABLE\_INBOUND\_NOTIFICATIONS:** This value is a **DWORD** used as an on/off switch. If this value is off, the firewall MAY display a notification to the user when an application is blocked from listening on a port.[<9>](#Appendix_A_9" \o "Product behavior note 9) If this value is on, the firewall MUST NOT display such a notification. The merge law for this option is to let the value of the **GroupPolicyRSoPStore** win if it is configured; otherwise, the local store value is used. This symbolic constant has a value of 10.

**FW\_PROFILE\_CONFIG\_AUTH\_APPS\_ALLOW\_USER\_PREF\_MERGE:** This value is a **DWORD** used as an on/off switch. If this value is off, authorized application firewall rules in the local store are ignored and not enforced. The merge law for this option is to let the value of the **GroupPolicyRSoPStore** win if it is configured; otherwise, the local store value is used. This symbolic constant has a value of 11.

The authorized application firewall rules consist of the [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b) objects where all of the following are true:

**wszLocalApplication** is not NULL

**wszLocalService** == NULL

(**wIpProtocol** == 6) || (**wIpProtocol** == 17)

**LocalPorts.Ports.dwNumEntries** == 0

**LocalPorts.wPortKeywords** == FW\_PORT\_KEYWORD\_NONE

Note that for the **wIpProtocol** condition, the numbers 6 and 17 are the assigned Internet protocol numbers for [**TCP**](#gt_b08d36f6-b5c6-4ce4-8d2d-6f2ab75ea4cb) and UDP respectively (for more information, see [[IANA-PROTO-NUM]](https://go.microsoft.com/fwlink/?LinkId=89889)).

**FW\_PROFILE\_CONFIG\_GLOBAL\_PORTS\_ALLOW\_USER\_PREF\_MERGE:** This value is a **DWORD** used as an on/off switch. If this value is off, global port firewall rules in the local store are ignored and not enforced. The setting only has meaning if it is set or enumerated in the [**Group Policy**](#gt_defe8c22-1365-4e5e-abf7-46ad112d3bda) store or if it is enumerated from the **GroupPolicyRSoPStore**. The merge law for this option is to let the value **GroupPolicyRSoPStore** win if it is configured; otherwise, the local store value is used. This symbolic constant has a value of 12.

The global port firewall rules consist of the FW\_RULE objects where all of the following are true:

**wszLocalApplication** == NULL

**wszLocalService** == NULL

(**wIpProtocol** == 6) || (**wIpProtocol** == 17)

**LocalPorts.Ports.dwNumEntries** == 1

**LocalPorts.wPortKeywords** == FW\_PORT\_KEYWORD\_NONE

Note that for the **wIpProtocol** condition, the numbers 6 and 17 are the assigned Internet protocol numbers for TCP and UDP respectively (for more information, see [IANA-PROTO-NUM]).

**FW\_PROFILE\_CONFIG\_ALLOW\_LOCAL\_POLICY\_MERGE:** This value is a **DWORD** used as an on/off switch. If this value is off, firewall rules from the local store are ignored and not enforced. The merge law for this option is to always use the value of the **GroupPolicyRSoPStore**. This value is valid for all schema versions. This symbolic constant has a value of 13.

**FW\_PROFILE\_CONFIG\_ALLOW\_LOCAL\_IPSEC\_POLICY\_MERGE:** This value is a **DWORD**; it is an on/off switch. If this value is off, connection security rules from the local store are ignored and not enforced, regardless of the schema version and connection security rule version. The merge law for this option is to always use the value of the **GroupPolicyRSoPStore**. This symbolic constant has a value of 14.

**FW\_PROFILE\_CONFIG\_DISABLED\_INTERFACES:** This value is an [FW\_INTERFACE\_LUIDS](#Section_ea420d0f03ed48e5b786621db56419d5) structure that represents the network adapters where the firewall (only the firewall rules and actions) is off. The merge law for this option is to let the value of the **GroupPolicyRSoPStore** win if it is configured; otherwise, the local store value is used. This symbolic constant has a value of 15.

**FW\_PROFILE\_CONFIG\_DEFAULT\_OUTBOUND\_ACTION:** This value is the action that the firewall does by default (and evaluates at the very end) on outbound connections. The allow action is represented by 0x00000000; 0x00000001 represents a block action. The merge law for this option is to let the value of the **GroupPolicyRSoPStore** win if it is configured; otherwise, the local store value is used. This symbolic constant has a value of 16.

**FW\_PROFILE\_CONFIG\_DEFAULT\_INBOUND\_ACTION:** This value is the action that the firewall does by default (and evaluates at the very end) on inbound connections. The allow action is represented by 0x00000000; 0x00000001 represents a block action. The merge law for this option is to let the value of the **GroupPolicyRSoPStore**.win if it is configured; otherwise, the local store value is used. This symbolic constant has a value of 17.

**FW\_PROFILE\_CONFIG\_DISABLE\_STEALTH\_MODE\_IPSEC\_SECURED\_PACKET\_EXEMPTION:** This value is a **DWORD** used as an on/off switch. This option is ignored if **FW\_PROFILE\_CONFIG\_DISABLE\_STEALTH\_MODE** is on. Otherwise, when this option is on, the firewall's stealth mode rules MUST NOT prevent the host computer from responding to unsolicited network traffic if that traffic is secured by [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb). The merge law for this option is to let the value of the **GroupPolicyRSoPStore** win if it is configured; otherwise, the local store value is used. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used. This symbolic constant has a value of 18.

**FW\_PROFILE\_CONFIG\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 19.

### FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_VALUES

This enumeration identifies specific traffic to be exempted from performing [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb).

1. typedef enum \_FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_VALUES
2. {
3. FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_NONE = 0x0000,
4. FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_NEIGHBOR\_DISC = 0x0001,
5. FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_ICMP = 0x0002,
6. FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_ROUTER\_DISC = 0x0004,
7. FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_DHCP = 0x0008,
8. FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_MAX = 0x0010,
9. FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_MAX\_V2\_0 = 0x0004
10. } FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_VALUES;

**FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_NONE:** No IPsec exemptions.

**FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_NEIGHBOR\_DISC:** Exempt neighbor discover IPv6 ICMP type-codes from IPsec.

**FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_ICMP:** Exempt ICMP from IPsec.

**FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_ROUTER\_DISC:** Exempt router discover IPv6 ICMP type-codes from IPsec.

**FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_DHCP:** Exempt both IPv4 and IPv6 DHCP traffic from IPsec.

**FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_MAX:**  This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 0x0010.

**FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_MAX\_V2\_0:** This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x0200 and earlier. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 0x0004.

### FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING\_VALUES

This enumeration is used to describe how preshared keys are encoded before being used.

1. typedef enum \_FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING\_VALUES
2. {
3. FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING\_NONE = 0,
4. FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING\_UTF\_8 = 1,
5. FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING\_MAX = 2
6. } FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING\_VALUES;

**FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING\_NONE:** Preshared key is not encoded. Instead, it is kept in its wide-character format. This symbolic constant has a value of 0.

**FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING\_UTF\_8:** Encode the preshared key using UTF-8. This symbolic constant has a value of 1.

**FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 2.

### FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_VALUES

This enumeration is used to describe when [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) [**security associations**](#gt_67cbf867-7a49-41f3-a68f-37b5f9035acb) can be established across NAT devices.

1. typedef enum \_FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_VALUES
2. {
3. FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_NEVER = 0,
4. FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_SERVER\_BEHIND\_NAT = 1,
5. FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_SERVER\_AND\_CLIENT\_BEHIND\_NAT = 2,
6. FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_MAX = 3
7. } FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_VALUES;

**FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_NEVER:** IPsec does not cross NAT boundaries. This symbolic constant has a value of 0.

**FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_SERVER\_BEHIND\_NAT:** IPsec security associations can be established when the server is across NAT boundaries. This symbolic constant has a value of 1.

**FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_SERVER\_AND\_CLIENT\_BEHIND\_NAT:** IPsec security associations can be established when the server and client are across NAT boundaries. This symbolic constant has a value of 2.

**FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 3.

### FW\_GLOBAL\_CONFIG

This enumeration identifies the global policy configuration options. Each configuration option has a merge law that is used to determine how to merge the values of these options across stores.

1. typedef enum \_tag\_FW\_GLOBAL\_CONFIG
2. {
3. FW\_GLOBAL\_CONFIG\_INVALID = 0,
4. FW\_GLOBAL\_CONFIG\_POLICY\_VERSION\_SUPPORTED = 1,
5. FW\_GLOBAL\_CONFIG\_CURRENT\_PROFILE = 2,
6. FW\_GLOBAL\_CONFIG\_DISABLE\_STATEFUL\_FTP = 3,
7. FW\_GLOBAL\_CONFIG\_DISABLE\_STATEFUL\_PPTP = 4,
8. FW\_GLOBAL\_CONFIG\_SA\_IDLE\_TIME = 5,
9. FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING = 6,
10. FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT = 7,
11. FW\_GLOBAL\_CONFIG\_CRL\_CHECK = 8,
12. FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT = 9,
13. FW\_GLOBAL\_CONFIG\_POLICY\_VERSION = 10,
14. FW\_GLOBAL\_CONFIG\_BINARY\_VERSION\_SUPPORTED = 11,
15. FW\_GLOBAL\_CONFIG\_IPSEC\_TUNNEL\_REMOTE\_MACHINE\_AUTHORIZATION\_LIST = 12,
16. FW\_GLOBAL\_CONFIG\_IPSEC\_TUNNEL\_REMOTE\_USER\_AUTHORIZATION\_LIST = 13,
17. FW\_GLOBAL\_CONFIG\_OPPORTUNISTICALLY\_MATCH\_AUTH\_SET\_PER\_KM = 14,
18. FW\_GLOBAL\_CONFIG\_IPSEC\_TRANSPORT\_REMOTE\_MACHINE\_AUTHORIZATION\_LIST = 15,
19. FW\_GLOBAL\_CONFIG\_IPSEC\_TRANSPORT\_REMOTE\_USER\_AUTHORIZATION\_LIST = 16,
20. FW\_GLOBAL\_CONFIG\_ENABLE\_PACKET\_QUEUE = 17,
21. FW\_GLOBAL\_CONFIG\_MAX = 18
22. } FW\_GLOBAL\_CONFIG;

**FW\_GLOBAL\_CONFIG\_INVALID:** This value MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 0.

**FW\_GLOBAL\_CONFIG\_POLICY\_VERSION\_SUPPORTED:** This value is a DWORD containing the maximum policy version that the server host can accept. The version number is two octets in size. The lowest-order octet is the minor version; the second-to-lowest octet is the major version. This value is not merged and is always a fixed value for a particular firewall and advanced security components software build. This symbolic constant has a value of 1.

**FW\_GLOBAL\_CONFIG\_CURRENT\_PROFILE:** This value is a DWORD and contains a bitmask of the current enforced profiles that are maintained by the server firewall host. See [FW\_PROFILE\_TYPE (section 2.2.2)](#Section_7704e238174d4a5eb8095f3787dd8acc) for the bitmasks that are used to identify profile types. This value is available only in the dynamic store; therefore, it is not merged and has no merge law. This symbolic constant has a value of 2.

**FW\_GLOBAL\_CONFIG\_DISABLE\_STATEFUL\_FTP:** This value is an on/off switch. If off, the firewall performs stateful File Transfer Protocol (FTP) filtering to allow secondary connections. The value is a DWORD; 0x00000000 means off; 0x00000001 means on. The merge law for this option is to let "on" values win. This symbolic constant has a value of 3.

**FW\_GLOBAL\_CONFIG\_DISABLE\_STATEFUL\_PPTP:** This value is an on/off switch. If off, the firewall performs stateful Point-to-Point Tunneling Protocol (PPTP) analysis. The value is a DWORD; 0x00000000 means off; 0x00000001 means on. The merge law for this option is to let "on" values win. This symbolic constant has a value of 4.

**FW\_GLOBAL\_CONFIG\_SA\_IDLE\_TIME:** This value configures the [**security association**](#gt_67cbf867-7a49-41f3-a68f-37b5f9035acb) idle time, in seconds. Security associations are deleted after network traffic is not seen for this specified period of time. The value is a DWORD and MUST be a value in the range of 300 to 3,600 inclusive. The merge law for this option is to let the value of the **GroupPolicyRSoPStore** win if it is configured; otherwise, use the local store value. This symbolic constant has a value of 5.

**FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING:** This configuration value specifies the preshared key encoding that is used. The value is a DWORD and MUST be a valid value from the [FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING\_VALUES](#Section_b9d24a5e77554c60adebe0c7a718f909) enumeration. The merge law for this option is to let the value of the **GroupPolicyRSoPStore** win if it is configured; otherwise, use the local store value. This symbolic constant has a value of 6.

**FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT:** This configuration value configures [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) exceptions. The value is a DWORD and MUST be a combination of the valid flags that are defined in [FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_VALUES](#Section_7daabd9f74c34295add6e2402b01b191); therefore, the maximum value MUST always be FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_MAX-1 for servers supporting a schema version of 0x0201 and FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_MAX\_V2\_0-1 for servers supporting a schema version of 0x0200. If the maximum value is exceeded when the method RRPC\_FWSetGlobalConfig (Opnum 4) is called, the method returns ERROR\_INVALID\_PARAMETER. This error code is returned if no other preceding error is discovered. The merge law for this option is to let the value of the **GroupPolicyRSoPStore** win if it is configured; otherwise, use the local store value. This symbolic constant has a value of 7.

**FW\_GLOBAL\_CONFIG\_CRL\_CHECK:** This value specifies how [**certificate revocation list (CRL)**](#gt_4f22841f-249b-42fb-a31a-5049c00be939) verification is enforced. The value is a DWORD and MUST be 0, 1, or 2. A value of 0 disables CRL checking. A value of 1 specifies that CRL checking is attempted and that certificate validation fails only if the certificate is revoked. Other failures that are encountered during CRL checking (such as the revocation URL being unreachable) do not cause certificate validation to fail. A value of 2 means that checking is required and that certificate validation fails if any error is encountered during CRL processing. The merge law for this option is to let the value of the **GroupPolicyRSoPStore** win if it is configured; otherwise, use the local store value. This symbolic constant has a value of 8.

**FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT:** This value is configured when an IPsec security association can be established with a computer across NAT devices. The value is of type [FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_VALUES](#Section_69a0da19647041958a32a204566fa813) and MUST contain valid values of the same enumeration type. The merge law for this option is to let the value of the **GroupPolicyRSoPStore** win if it is configured; otherwise, use the local store value. This symbolic constant has a value of 9.

**FW\_GLOBAL\_CONFIG\_POLICY\_VERSION:** This value contains the policy version of the policy store being managed. This value is not merged and therefore, has no merge law. This symbolic constant has a value of 10.

**FW\_GLOBAL\_CONFIG\_BINARY\_VERSION\_SUPPORTED:** This value contains the binary version of the structures and data types that are supported by the server. This value is not merged. In addition, this value is always a fixed value for a specific firewall and advanced security component's software build. This symbolic constant has a value of 11. This value identifies a policy configuration option that is supported only on servers that have a schema version of 0x0201.

**FW\_GLOBAL\_CONFIG\_IPSEC\_TUNNEL\_REMOTE\_MACHINE\_AUTHORIZATION\_LIST:** This value represents a list of remote machines that are allowed to send and receive traffic through the tunnels which request this access check. Machines in the list are allowed through the tunnels. Machines not in the list are denied through the tunnels. The list is specified as a security descriptor which specifies which [**SIDs**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) ([[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.2.1) of the remote machines. The value is a [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string in Security Descriptor Definition Language (SDDL) format ([MS-DTYP] section 2.5.1). This symbolic constant has a value of 12.

**FW\_GLOBAL\_CONFIG\_IPSEC\_TUNNEL\_REMOTE\_USER\_AUTHORIZATION\_LIST:** This value represents a list of remote users who are allowed to send and receive traffic through the tunnels which request this access check. Users in the list are allowed through the tunnels. Users not in the list are denied through the tunnels. The list is specified as a security descriptor which specifies which SIDs ([MS-DTYP] section 2.4.2.1) of the remote users. The value is a Unicode string in SDDL format ([MS-DTYP] section 2.5.1). This symbolic constant has a value of 13.

**FW\_GLOBAL\_CONFIG\_OPPORTUNISTICALLY\_MATCH\_AUTH\_SET\_PER\_KM:** This value is a DWORD used as an on/off switch. When this option is off, keying modules MUST ignore the entire authentication set if they do not support all of the authentication suites specified in the set. When this option is on, keying modules MUST ignore only the authentication suites that they don’t support. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used. This symbolic constant has a value of 14.

**FW\_GLOBAL\_CONFIG\_IPSEC\_TRANSPORT\_REMOTE\_MACHINE\_AUTHORIZATION\_LIST:** This value is a Unicode string in Security Descriptor Definition Language (SDDL) format ([MS-DTYP] section 2.5.1). The security descriptor describes which remote machines are allowed to send and receive traffic secured by transport mode connection security rules which request this access check. Machines granted access by the security descriptor are allowed to send and receive traffic. Machines denied access by the security descriptor are blocked from sending and receiving traffic. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used. This symbolic constant has a value of 15.

**FW\_GLOBAL\_CONFIG\_IPSEC\_TRANSPORT\_REMOTE\_USER\_AUTHORIZATION\_LIST:** This value is a Unicode string in Security Descriptor Definition Language (SDDL) format. The security descriptor describes which remote users are allowed to send and receive traffic secured by transport mode connection security rules which request this access check. Users granted access by the security descriptor are allowed to send and receive traffic. Users denied access by the security descriptor are blocked from sending and receiving traffic. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used. This symbolic constant has a value of 16.

**FW\_GLOBAL\_CONFIG\_ENABLE\_PACKET\_QUEUE:** This value specifies how scaling for the software on the receive side is enabled for both the encrypted receive and clear text forward path for the IPsec tunnel gateway scenario (as configured by [FW\_CS\_RULE (section 2.2.54)](#Section_0d0641105f2e4b68aa63032c6cd5e4c6)). Use of this option also ensures that the packet order is preserved. The data type for this option value is a DWORD and is a combination of flags. A value of 0x00 indicates that all queuing is to be disabled. A value of 0x01 specifies that inbound encrypted packets are to be queued. A value of 0x02 specifies that packets are to be queued after decryption is performed for forwarding. This symbolic constant has a value of 17.

**FW\_GLOBAL\_CONFIG\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. This symbolic constant is defined for simplicity in writing IDL definitions and code. It has a value of 18.

**Note** The value of **FW\_GLOBAL\_CONFIG\_MAX** depends of the number of members in this enumeration, which, in turn, depends on the schema version. See the descriptions of the previous enumeration members to determine what this value is for each schema version.

### FW\_CONFIG\_FLAGS

This enumeration identifies flags that can be set on the RRPC\_FWGetConfig (Opnum 10) and RRPC\_FWGetGlobalConfig (Opnum 3) methods.

1. typedef enum \_FW\_CONFIG\_FLAGS
2. {
3. FW\_CONFIG\_FLAG\_RETURN\_DEFAULT\_IF\_NOT\_FOUND = 0x0001
4. } FW\_CONFIG\_FLAGS;

**FW\_CONFIG\_FLAG\_RETURN\_DEFAULT\_IF\_NOT\_FOUND:** If this flag is specified, and if the RRPC\_FWGetConfig (Opnum 10) method or the RRPC\_FWGetGlobalConfig (Opnum 3) method fails to find the configuration value in the policy store, then the call will succeed and return the default value used by the firewall service. If this flag is not specified, these methods will fail with ERROR\_FILE\_NOT\_FOUND. The default set of values returned by these two calls is a firewall and advanced security component implementation-specific[<10>](#Appendix_A_10" \o "Product behavior note 10) decision, and is outside the scope of this protocol specification.

### FW\_NETWORK

This structure represents a network that is associated with a firewall profile. It is used for display purposes in user interfaces.

1. typedef struct \_tag\_FW\_NETWORK {
2. [string, unique] wchar\_t\* pszName;
3. FW\_PROFILE\_TYPE ProfileType;
4. } FW\_NETWORK,
5. \*PFW\_NETWORK;

**pszName:**  A pointer to a [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string that represents the name of the network.

**ProfileType:**  The profile type that is associated with the network. The type MUST be one of the [FW\_PROFILE\_TYPE](#Section_7704e238174d4a5eb8095f3787dd8acc) flags, except FW\_PROFILE\_TYPE\_ALL.

### FW\_ADAPTER

This structure represents a network interface in the host. It is used for display purposes in the user interface when configuring the **FW\_PROFILE\_CONFIG\_DISABLED\_INTERFACES** (section [2.2.37](#Section_5a6e0d39802d456bb483c7360566fcdd)) configuration option.

1. typedef struct \_tag\_FW\_ADAPTER {
2. [string, unique] wchar\_t\* pszFriendlyName;
3. GUID Guid;
4. } FW\_ADAPTER,
5. \*PFW\_ADAPTER;

**pszFriendlyName:**  A pointer to a [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string that presents the friendly name that is associated with the network interface.

**Guid:**  A [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) that uniquely identifies the interface in the host system.

### FW\_DIAG\_APP

This structure is not used on the wire.

### FW\_RULE\_CATEGORY

This enumeration represents the classes of functionality that a third-party software component can register for, take ownership of, and commit to implement. The implementation of such functionality by the firewall and advanced security component, or by the third-party software component, are implementation-specific decisions. This enumeration is only used to present the state of the registrations.

1. typedef [v1\_enum] enum \_tag\_FW\_RULE\_CATEGORY
2. {
3. FW\_RULE\_CATEGORY\_BOOT = 0,
4. FW\_RULE\_CATEGORY\_STEALTH = 1,
5. FW\_RULE\_CATEGORY\_FIREWALL = 2,
6. FW\_RULE\_CATEGORY\_CONSEC = 3,
7. FW\_RULE\_CATEGORY\_MAX = 4
8. } FW\_RULE\_CATEGORY,
9. \*PFW\_RULE\_CATEGORY;

**FW\_RULE\_CATEGORY\_BOOT:** This category of functionality represents the policy that is used while the system is starting up and the firewall and advance security component is not yet running. This symbolic constant has a value of 0.

**FW\_RULE\_CATEGORY\_STEALTH:** This category of functionality represents the policy that is used to make the system appear invisible when it is connected to a network. For example, this functionality helps prevent attackers from discovering the host and the ports that open to the host. This symbolic constant has a value of 1.

**FW\_RULE\_CATEGORY\_FIREWALL:** This category of functionality represents functions that are performed by firewall objects while they are present on the FW\_STORE\_TYPE\_LOCAL, FW\_STORE\_TYPE\_DYNAMIC, and FW\_STORE\_TYPE\_GP\_RSOP policy stores (see section [2.2.1](#Section_37ebed958abf472c8b4b7a510a2a6baa)). This symbolic constant has a value of 2.

**FW\_RULE\_CATEGORY\_CONSEC:** This category of functionality represents functions that are performed by the connection security objects. This symbolic constant has a value of 3.

**FW\_RULE\_CATEGORY\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 4.

### FW\_PRODUCT

This structure represents a third-party software component that registers with the firewall and advanced security component to implement some of the categories.

1. typedef struct \_tag\_FW\_PRODUCT {
2. DWORD dwFlags;
3. DWORD dwNumRuleCategories;
4. [size\_is(dwNumRuleCategories), unique]
5. FW\_RULE\_CATEGORY\* pRuleCategories;
6. [string, ref] wchar\_t\* pszDisplayName;
7. [string, unique] wchar\_t\* pszPathToSignedProductExe;
8. } FW\_PRODUCT,
9. \*PFW\_PRODUCT;

**dwFlags:**  This field is not used.

**dwNumRuleCategories:**  The number of rule categories with which the third-party software component registered.

**pRuleCategories:**  A pointer to an array of **dwNumRuleCategories** that are contiguous [FW\_RULE\_CATEGORY](#Section_16b2c1a3789f4cdeb3d1ecd9182b60fe) elements.

**pszDisplayName:**  A pointer to a [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string. The string represents the name of the third-party software component.

**pszPathToSignedProductExe:**  A pointer to a Unicode string. The string represents the file path to the binary executable of the third-party software component.

### FW\_IP\_VERSION

This enumeration is used to represent the two current IP protocol versions in use: IP version 4 and IP version 6.

1. typedef enum \_tag\_FW\_IP\_VERSION
2. {
3. FW\_IP\_VERSION\_INVALID = 0,
4. FW\_IP\_VERSION\_V4,
5. FW\_IP\_VERSION\_V6 = 2,
6. FW\_IP\_VERSION\_MAX = 3
7. } FW\_IP\_VERSION;

**FW\_IP\_VERSION\_INVALID:** This value MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 0.

**FW\_IP\_VERSION\_V4:** This value represents IPv4. This symbolic constant has a value of 1.

**FW\_IP\_VERSION\_V6:** This value represents the IPv6. This symbolic constant has a value of 2.

**FW\_IP\_VERSION\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 3.

### FW\_IPSEC\_PHASE

This enumeration is used to identify the [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) phase of negotiations.

1. typedef enum \_tag\_FW\_IPSEC\_PHASE
2. {
3. FW\_IPSEC\_PHASE\_INVALID = 0,
4. FW\_IPSEC\_PHASE\_1 = 1,
5. FW\_IPSEC\_PHASE\_2 = 2,
6. FW\_IPSEC\_PHASE\_MAX = 3
7. } FW\_IPSEC\_PHASE;

**FW\_IPSEC\_PHASE\_INVALID:** This value MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 0.

**FW\_IPSEC\_PHASE\_1:** This value represents the IPsec first phase of negotiations, also called main mode. This symbolic constant has a value of 1.

**FW\_IPSEC\_PHASE\_2:** This value represents the IPsec second phase of negotiations. A phase 2 authentication is the second authentication and can mean extended mode or quick mode. On auth sets, phase 2 authentication refers to extended mode (specified in [[MS-AIPS]](%5bMS-AIPS%5d.pdf#Section_eee3de6438474451978e9513ff187d30) sections 3.6 and 3.7); and on crypto sets, phase 2 refers to quick mode (specified in [MS-AIPS] sections 3.4 and 3.5). This symbolic constant has a value of 2.

**FW\_IPSEC\_PHASE\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 3.

### FW\_CS\_RULE\_FLAGS

This enumeration describes flag values for connection security rules.

1. typedef enum \_tag\_FW\_CS\_RULE\_FLAGS
2. {
3. FW\_CS\_RULE\_FLAGS\_NONE = 0x00,
4. FW\_CS\_RULE\_FLAGS\_ACTIVE = 0x01,
5. FW\_CS\_RULE\_FLAGS\_DTM = 0x02,
6. FW\_CS\_RULE\_TUNNEL\_BYPASS\_IF\_ENCRYPTED = 0x08,
7. FW\_CS\_RULE\_OUTBOUND\_CLEAR = 0x10,
8. FW\_CS\_RULE\_FLAGS\_APPLY\_AUTHZ = 0x20,
9. FW\_CS\_RULE\_FLAGS\_KEY\_MANAGER\_ALLOW\_DICTATE\_KEY = 0x40,
10. FW\_CS\_RULE\_FLAGS\_KEY\_MANAGER\_ALLOW\_NOTIFY\_KEY = 0x80,
11. FW\_CS\_RULE\_FLAGS\_SECURITY\_REALM = 0x100,
12. FW\_CS\_RULE\_FLAGS\_MAX = 0x200,
13. FW\_CS\_RULE\_FLAGS\_MAX2\_1 = 0x02,
14. FW\_CS\_RULE\_FLAGS\_MAX\_V2\_10 = 0x40,
15. FW\_CS\_RULE\_FLAGS\_MAX\_V2\_20 = 0x100
16. } FW\_CS\_RULE\_FLAGS;

**FW\_CS\_RULE\_FLAGS\_NONE:** This value means that none of the following flags are set. This value is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code.

**FW\_CS\_RULE\_FLAGS\_ACTIVE:** If this flag is set, the rule is enabled; otherwise, the rule is disabled.

**FW\_CS\_RULE\_FLAGS\_DTM:** If this flag is set, the rule is a dynamic tunnel mode rule.

**FW\_CS\_RULE\_TUNNEL\_BYPASS\_IF\_ENCRYPTED:** This flag MUST only be set on tunnel mode rules. If this flag is set and traffic is already arriving encrypted, it is exempted from the tunnel.

**FW\_CS\_RULE\_OUTBOUND\_CLEAR:** This flag MUST only be set on tunnel mode rules. If set, when outbound traffic matches the rule, it leaves unprotected, but inbound traffic MUST arrive through the tunnel.

**FW\_CS\_RULE\_FLAGS\_APPLY\_AUTHZ:** This flag MUST only be set on tunnel mode rules. If this flag is set, the authenticated peers of the traffic MUST match the SDDLs that are specified in FW\_GLOBAL\_CONFIG\_IPSEC\_TUNNEL\_REMOTE\_MACHINE\_AUTHORIZATION\_LIST and FW\_GLOBAL\_CONFIG\_IPSEC\_TUNNEL\_REMOTE\_USER\_AUTHORIZATION\_LIST.

**FW\_CS\_RULE\_FLAGS\_KEY\_MANAGER\_ALLOW\_DICTATE\_KEY:** If this flag is set, external key managers are permitted to dictate the cryptographic keys used. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used.

**FW\_CS\_RULE\_FLAGS\_KEY\_MANAGER\_ALLOW\_NOTIFY\_KEY:** If this flag is set, external key managers are notified of the cryptographic keys used. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used.

**FW\_CS\_RULE\_FLAGS\_SECURITY\_REALM:** If this flag is set, the connection security rule is associated with a security realm. The **wszRuleId** of the connection security rule is the same as the IPsec Security Realm ID that it is associated with. For schema versions 0x0200, 0x0201, 0x20A, and 0x0214, this value is invalid and MUST NOT be used.

**FW\_CS\_RULE\_FLAGS\_MAX:** This value and values that exceed this value are not valid for all schema versions and MUST NOT be used. It is only defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 0x200.

**FW\_CS\_RULE\_FLAGS\_MAX2\_1:** This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x0201 and earlier. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 0x02.

**FW\_CS\_RULE\_FLAGS\_MAX\_V2\_10:** This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x020A and earlier. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 0x40.

**FW\_CS\_RULE\_FLAGS\_MAX\_V2\_20:** This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x0214 and earlier. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 0x100.

### FW\_CS\_RULE\_ACTION

This enumeration identifies the possible actions a connection security rule (section [2.2.54](#Section_0d0641105f2e4b68aa63032c6cd5e4c6)) can have.

1. typedef enum \_tag\_FW\_CS\_RULE\_ACTION
2. {
3. FW\_CS\_RULE\_ACTION\_INVALID = 0,
4. FW\_CS\_RULE\_ACTION\_SECURE\_SERVER = 1,
5. FW\_CS\_RULE\_ACTION\_BOUNDARY = 2,
6. FW\_CS\_RULE\_ACTION\_SECURE = 3,
7. FW\_CS\_RULE\_ACTION\_DO\_NOT\_SECURE = 4,
8. FW\_CS\_RULE\_ACTION\_MAX = 5
9. } FW\_CS\_RULE\_ACTION;

**FW\_CS\_RULE\_ACTION\_INVALID:** This value MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 0.

**FW\_CS\_RULE\_ACTION\_SECURE\_SERVER:** This action requires inbound traffic to be [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) traffic and attempts to secure outbound traffic with IPsec. This symbolic constant has a value of 1.

**FW\_CS\_RULE\_ACTION\_BOUNDARY:** This action attempts to secure inbound and outbound traffic with IPsec. If the action fails to secure the traffic, the traffic still flows on the clear. This symbolic constant has a value of 2.

**FW\_CS\_RULE\_ACTION\_SECURE:** This action requires inbound and outbound traffic to be secured by IPsec. This symbolic constant has a value of 3.

**FW\_CS\_RULE\_ACTION\_DO\_NOT\_SECURE:** This action exempts the traffic from being secured by IPsec. This symbolic constant has a value of 4.

**FW\_CS\_RULE\_ACTION\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 5.

### FW\_CS\_RULE2\_10

This structure describes a connection security rule that is used by the 2.10 binary version for servers and clients (see sections [1.7](#Section_6b14b8f991ab4f09873f9e2334196cdc) and [2.2.37](#Section_5a6e0d39802d456bb483c7360566fcdd)). The fields of this structure are identical to the [FW\_CS\_RULE](#Section_0d0641105f2e4b68aa63032c6cd5e4c6) structure, and their meanings are covered in section 2.2.54.

1. typedef struct \_tag\_FW\_CS\_RULE2\_10 {
2. struct \_tag\_FW\_CS\_RULE2\_10\* pNext;
3. unsigned short wSchemaVersion;
4. [string, range(1,10001), ref] wchar\_t\* wszRuleId;
5. [string, range(1,10001)] wchar\_t\* wszName;
6. [string, range(1,10001)] wchar\_t\* wszDescription;
7. unsigned long dwProfiles;
8. FW\_ADDRESSES Endpoint1;
9. FW\_ADDRESSES Endpoint2;
10. FW\_INTERFACE\_LUIDS LocalInterfaceIds;
11. unsigned long dwLocalInterfaceTypes;
12. unsigned long dwLocalTunnelEndpointV4;
13. unsigned char LocalTunnelEndpointV6[16];
14. unsigned long dwRemoteTunnelEndpointV4;
15. unsigned char RemoteTunnelEndpointV6[16];
16. FW\_PORTS Endpoint1Ports;
17. FW\_PORTS Endpoint2Ports;
18. [range(0,256)] unsigned short wIpProtocol;
19. [string, range(1,10001)] wchar\_t\* wszPhase1AuthSet;
20. [string, range(1,10001)] wchar\_t\* wszPhase2CryptoSet;
21. [string, range(1,10001)] wchar\_t\* wszPhase2AuthSet;
22. [range(FW\_CS\_RULE\_ACTION\_SECURE\_SERVER, FW\_CS\_RULE\_ACTION\_MAX)]
23. FW\_CS\_RULE\_ACTION Action;
24. unsigned short wFlags;
25. [string, range(1,10001)] wchar\_t\* wszEmbeddedContext;
26. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
27. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX-1)]
28. FW\_RULE\_ORIGIN\_TYPE Origin;
29. [string, range(1,10001)] wchar\_t\* wszGPOName;
30. FW\_RULE\_STATUS Status;
31. [string, range(1,512)] wchar\_t\* wszMMParentRuleId;
32. unsigned long Reserved;
33. [size\_is((Reserved & FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA) ? 1 : 0)]
34. PFW\_OBJECT\_METADATA pMetaData;
35. } FW\_CS\_RULE2\_10,
36. \*PFW\_CS\_RULE2\_10;

### FW\_CS\_RULE2\_0

This structure describes a connection security rule that is used by the 2.0 binary version for servers and clients (see sections [1.7](#Section_6b14b8f991ab4f09873f9e2334196cdc) and [2.2.37](#Section_5a6e0d39802d456bb483c7360566fcdd)). The fields of this structure are identical to the [FW\_CS\_RULE](#Section_0d0641105f2e4b68aa63032c6cd5e4c6) structure and their meanings are covered in section 2.2.54.

1. typedef struct \_tag\_FW\_CS\_RULE2\_0 {
2. struct \_tag\_FW\_CS\_RULE2\_0\* pNext;
3. unsigned short wSchemaVersion;
4. [string, range(1,10001), ref] wchar\_t\* wszRuleId;
5. [string, range(1,10001)] wchar\_t\* wszName;
6. [string, range(1,10001)] wchar\_t\* wszDescription;
7. unsigned long dwProfiles;
8. FW\_ADDRESSES Endpoint1;
9. FW\_ADDRESSES Endpoint2;
10. FW\_INTERFACE\_LUIDS LocalInterfaceIds;
11. unsigned long dwLocalInterfaceTypes;
12. unsigned long dwLocalTunnelEndpointV4;
13. unsigned char LocalTunnelEndpointV6[16];
14. unsigned long dwRemoteTunnelEndpointV4;
15. unsigned char RemoteTunnelEndpointV6[16];
16. FW\_PORTS Endpoint1Ports;
17. FW\_PORTS Endpoint2Ports;
18. [range(0,256)] unsigned short wIpProtocol;
19. [string, range(1,10001)] wchar\_t\* wszPhase1AuthSet;
20. [string, range(1,10001)] wchar\_t\* wszPhase2CryptoSet;
21. [string, range(1,10001)] wchar\_t\* wszPhase2AuthSet;
22. [range(FW\_CS\_RULE\_ACTION\_SECURE\_SERVER, FW\_CS\_RULE\_ACTION\_MAX - 1)]
23. FW\_CS\_RULE\_ACTION Action;
24. unsigned short wFlags;
25. [string, range(1,10001)] wchar\_t\* wszEmbeddedContext;
26. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
27. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX-1)]
28. FW\_RULE\_ORIGIN\_TYPE Origin;
29. [string, range(1,10001)] wchar\_t\* wszGPOName;
30. FW\_RULE\_STATUS Status;
31. } FW\_CS\_RULE2\_0,
32. \*PFW\_CS\_RULE2\_0;

### FW\_CS\_RULE

This structure describes a connection security rule.

1. typedef struct \_tag\_FW\_CS\_RULE {
2. struct \_tag\_FW\_CS\_RULE\* pNext;
3. unsigned short wSchemaVersion;
4. [string, range(1,10001), ref] wchar\_t\* wszRuleId;
5. [string, range(1,10001)] wchar\_t\* wszName;
6. [string, range(1,10001)] wchar\_t\* wszDescription;
7. unsigned long dwProfiles;
8. FW\_ADDRESSES Endpoint1;
9. FW\_ADDRESSES Endpoint2;
10. FW\_INTERFACE\_LUIDS LocalInterfaceIds;
11. unsigned long dwLocalInterfaceTypes;
12. unsigned long dwLocalTunnelEndpointV4;
13. unsigned char LocalTunnelEndpointV6[16];
14. unsigned long dwRemoteTunnelEndpointV4;
15. unsigned char RemoteTunnelEndpointV6[16];
16. FW\_PORTS Endpoint1Ports;
17. FW\_PORTS Endpoint2Ports;
18. [range(0,256)] unsigned short wIpProtocol;
19. [string, range(1,10001)] wchar\_t\* wszPhase1AuthSet;
20. [string, range(1,10001)] wchar\_t\* wszPhase2CryptoSet;
21. [string, range(1,10001)] wchar\_t\* wszPhase2AuthSet;
22. [range(FW\_CS\_RULE\_ACTION\_SECURE\_SERVER, FW\_CS\_RULE\_ACTION\_MAX - 1)]
23. FW\_CS\_RULE\_ACTION Action;
24. unsigned short wFlags;
25. [string, range(1,10001)] wchar\_t\* wszEmbeddedContext;
26. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
27. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX-1)]
28. FW\_RULE\_ORIGIN\_TYPE Origin;
29. [string, range(1,10001)] wchar\_t\* wszGPOName;
30. FW\_RULE\_STATUS Status;
31. [string, range(1,512)] WCHAR\* wszMMParentRuleId;
32. DWORD Reserved;
33. [size\_is((Reserved & FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA) ? 1 : 0)]
34. PFW\_OBJECT\_METADATA pMetaData;
35. [string, range(1,512)] WCHAR\* wszRemoteTunnelEndpointFqdn;
36. FW\_ADDRESSES RemoteTunnelEndpoints;
37. DWORD dwKeyModules;
38. DWORD FwdPathSALifetime;
39. [string, range(1,10001)] LPWSTR\* wszTransportMachineAuthzSDDL;
40. [string, range(1,10001)] LPWSTR\* wszTransportUserAuthzSDDL;
41. } FW\_CS\_RULE,
42. \*PFW\_CS\_RULE;

**pNext:**  A pointer to the next FW\_CS\_RULE in the list.

**wSchemaVersion:**  Specifies the version of the rule.

**wszRuleId:**  A pointer to a [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string that uniquely identifies the rule.

**wszName:**  A pointer to a Unicode string that provides a friendly name for the rule.

**wszDescription:**  A pointer to a Unicode string that provides a friendly description for the rule.

**dwProfiles:**  A bitmask of the [FW\_PROFILE\_TYPE](#Section_7704e238174d4a5eb8095f3787dd8acc) flags. It is a condition that matches traffic on the specified profiles.

**Endpoint1:**  A condition that specifies the addresses of the first host of the traffic that the rule matches. An empty **EndPoint1** structure means that this condition is not applied (any match).

**Endpoint2:**  A condition that specifies the addresses of the second host of the traffic that the rule matches. An empty **EndPoint2** structure means that this condition is not applied (any match).

**LocalInterfaceIds:**  A condition that specifies the list of specific network interfaces that are used by the traffic that the rule matches. If the **LocalInterfaceIds** field does not specify an interface [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1), the rule applies to all interfaces; that is, the condition is not applied.

**dwLocalInterfaceTypes:**  A bitmask of [FW\_INTERFACE\_TYPE](#Section_d067c0284ea943199ddf0fb51c191b35). It is a condition that restricts the interface types used by the traffic that the rule matches. A value of 0x00000000 means the condition matches all interface types.

**dwLocalTunnelEndpointV4:**  This field specifies the IPv4 address of the [**endpoint**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee) that the host machines use as their local endpoint when [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) operates in tunnel mode.

**LocalTunnelEndpointV6:**  This field specifies the IPv6 address of the endpoint that the host machines use as their local endpoint when IPsec operates in tunnel mode.

**dwRemoteTunnelEndpointV4:**  This field specifies the IPv4 address of the endpoint that the host machines use as their remote endpoint when IPsec operates in tunnel mode.

**RemoteTunnelEndpointV6:**  This field specifies the IPv6 address of the endpoint that the host machines use as their remote endpoint when IPsec operates in tunnel mode.

**Endpoint1Ports:**  A condition that specifies the first host's ports of the [**TCP**](#gt_b08d36f6-b5c6-4ce4-8d2d-6f2ab75ea4cb) or UDP traffic that the rule matches.

**Endpoint2Ports:**  A condition that specifies the second host's ports of the TCP or UDP traffic that the rule matches.

**wIpProtocol:**  A condition that specifies the protocol of the traffic that the rule matches. If the value is in the range of 0 to 255, the value describes a protocol as in IETF IANA numbers (for more information, see [[IANA-PROTO-NUM]](https://go.microsoft.com/fwlink/?LinkId=89889)). If the value is 256, the rule matches any protocol.

**wszPhase1AuthSet:**  A Unicode string that represents the set identifier for the Phase1 authentication policy objects.

**wszPhase2CryptoSet:**  A Unicode string that represents the set identifier for the Phase2 cryptographic policy objects.

**wszPhase2AuthSet:**  A Unicode string that represents the set identifier of the Phase2 authentication policy objects. If this field is NULL, no second authentication is performed.

**Action:**  The connection security action that the rule takes for the traffic matches. This field MUST contain a valid value from the [FW\_CS\_RULE\_ACTION](#Section_358add6dadf74b6f8b6a66535ac802ed) enumeration.

**wFlags:**  A bit flag or flags from [FW\_CS\_RULE\_FLAGS](#Section_567fdc537b1c418b8b11dd7267f75bad).

**wszEmbeddedContext:**  A pointer to a Unicode string. It specifies a group name for this rule. Other components in the system use this string to enable or disable a group of rules by verifying that all rules have the same group name.

**PlatformValidityList:**  A condition in a rule that determines whether or not the rule is enforced by the local computer based on the local computer's platform information. The rule is enforced only if the local computer's operating system platform is an element of the set described by **PlatformValidityList**.[<11>](#Appendix_A_11" \o "Product behavior note 11)

**Origin:**  This field is the rule origin, as specified in the [FW\_RULE\_ORIGIN\_TYPE](#Section_9d295321d75c41c0ab0d7a78df40f77c) enumeration. It MUST be filled on enumerated rules and ignored on input.

**wszGPOName:**  A pointer to a Unicode string containing the displayName of the [**GPO**](#gt_dec32233-8776-4151-91a0-8624a2b9abb0) containing this object. When adding a new object, this field is not used. The client SHOULD set the value to NULL, and the server MUST ignore the value. When enumerating an existing object, if the client does not set the FW\_ENUM\_RULES\_FLAG\_RESOLVE\_GPO\_NAME flag, the server MUST set the value to NULL. Otherwise, the server MUST set the value to the displayName of the GPO containing the object or NULL if the object is not contained within a GPO. For details about how the server initializes an object from a GPO, see section [3.1.3](#Section_e8924ac5aa4a41d1bf654f46b3d399aa). For details about how the displayName of a GPO is stored, see [[MS-GPOL]](%5bMS-GPOL%5d.pdf#Section_62d1292462524052996f161d2b9019f4) section 2.3.

**Status:**  The status code of the rule, as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out when the structure is returned as output. On input, this field MUST be set to *FW\_RULE\_STATUS\_OK*.

**wszMMParentRuleId:**  This field is not used.

**Reserved:**  Not used other than to instruct [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) by using the FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA flag that a pointer to a FW\_OBJECT\_METADATA structure is present. It has no semantic meaning to the object itself

**pMetaData:**  A pointer to an FW\_OBJECT\_METADATA structure that contains specific metadata about the current state of the connection security rule.

**wszRemoteTunnelEndpointFqdn:**  A pointer to a Unicode string containing the [**fully qualified domain name (FQDN)**](#gt_1769aec9-237e-44ed-9014-1abb3ec6de6e) of the endpoints that the host machines use as their remote endpoint when IPsec operates in tunnel mode.

**RemoteTunnelEndpoints:**  This field specifies the IPv4 and IPv6 addresses of the endpoints that the host machines use as their remote endpoint when IPsec operates in tunnel mode.

**dwKeyModules:**  A bitmask of the FW\_KEY\_MODULE flags. It specifies the key modules used to establish the cryptographic keys used by IPsec.

**FwdPathSALifetime:**  This value is the lifetime in seconds before a Phase2 established key is renegotiated if the key is used to secure traffic forwarded from one interface to another on the same host machine.

**wszTransportMachineAuthzSDDL:**  A pointer to a Unicode string in Security Descriptor Definition Language (SDDL) format ([[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.2.36). The security descriptor describes which remote machines are allowed to send and receive traffic. Machines granted access by the security descriptor are allowed to send and receive traffic. Machines denied access by the security descriptor are blocked from sending and receiving traffic. This field MUST be null for tunnel mode rules.

**wszTransportUserAuthzSDDL:**  A pointer to a Unicode string in Security Descriptor Definition Language (SDDL) format ([MS-DTYP] section 2.2.36). The security descriptor describes which remote users are allowed to send and receive traffic. Users granted access by the security descriptor are allowed to send and receive traffic. Users denied access by the security descriptor are blocked from sending and receiving traffic. This field MUST be null for tunnel mode rules.

The following are semantic checks that connection security rules MUST pass:

* The **wSchemaVersion** field MUST NOT be less than 0x000200.
* The **wszRuleId** field MUST NOT contain the pipe '|' character, MUST NOT be NULL, MUST be a string of at least 1 character, and MUST NOT be greater than or equal to 512 characters.[<12>](#Appendix_A_12" \o "Product behavior note 12)
* The **wszName** field string MUST meet the following criteria:
  + MUST contain at least one character.
  + MUST contain less than 10,000 characters.
  + MUST NOT be NULL.
  + MUST NOT contain the pipe '|' character.
  + MUST NOT equal the string "ALL" (case-insensitive).
* If the **wszDescription** field string is not NULL, it MUST be at least 1 character, MUST NOT be greater than or equal to 10,000 characters, and MUST NOT contain the pipe '|' character.
* If the **wszEmbeddedContext** field string is not NULL, it MUST be at least 1 character, MUST NOT be greater than or equal to 10,000 characters, and MUST NOT contain the pipe '|' character.
* The **dwProfiles** field MUST NOT contain invalid values, and if it is not equal to the ALL profile type, it MUST NOT contain unknown profiles.
* The **wIpProtocol** field MUST NOT be greater than 256.
* If **wIpProtocol** is 6 or 17, the **wPortKeywords** field of **Endpoint1Ports** MUST be 0.
* If **wIpProtocol** is 6 or 17, the **wPortKeywords** field of **Endpoint2Ports** MUST be 0.
* If **wIpProtocol** is neither 6 nor 17, the **Endpoint1Ports** and **Endpoint2Ports** fields MUST be empty.
* If the **Endpoint1** field is not empty, **LocalInterfaceIds** MUST be empty and **dwLocalInterfaceTypes** MUST be 0. If the **Endpoint1** field is empty, **LocalInterfaceIds** MUST NOT be empty and **dwLocalInterfaceTypes** MUST NOT be 0.
* The **Endpoint1** and **Endpoint2** address keywords MUST contain valid address keywords.
* The **Endpoint1** and **Endpoint2** structures MUST NOT contain multicast v4 or v6 addresses.
* The **dwLocalInterfaceTypes** MUST NOT be greater than or equal to FW\_INTERFACE\_TYPE\_MAX.
* The **Action** field MUST be a valid action from the FW\_CS\_RULE\_ACTION enumeration.
* The **wFlags** field MUST NOT be greater than or equal to FW\_CS\_RULE\_FLAGS\_MAX.
* If the **Action** field is FW\_CS\_RULE\_ACTION\_DO\_NOT\_SECURE, **wszPhase1AuthSet**, **wszPhase2AuthSet**, and **wszPhase2CryptoSet** MUST all be NULL; otherwise, **wszPhase1AuthSet**, **wszPhase2AuthSet**, and **wszPhase2CryptoSet** MUST all be at least 1 character long, MUST NOT be greater than or equal to 1,000 characters,[<13>](#Appendix_A_13" \o "Product behavior note 13) and MUST NOT contain the pipe '|' character.

However, the **wszPhase2AuthSet** member can always be NULL. When **wszPhase2AuthSet** is not NULL, it SHOULD pass all of the string checks performed by the **wszPhase1AuthSet** member and the **wszPhase2CryptoSet** member.

* A tunnel rule has the **dwRemoteTunnelEndpointV4** (or V6) field as an address or the **dwLocalTunnelEndpointV4** (or V6) as an address. If the rule is a tunnel rule, the **Endpoint1** and **Endpoint2** addresses MUST NOT be empty; the **Action** field MUST be FW\_CS\_RULE\_ACTION\_SECURE; **wIpProtocol** MUST be ANY (256); **Endpoint1Ports** and **Endpoint2Ports** MUST be empty; and **dwRemoteTunnelEndpointV4** and **dwLocalTunnelEndpointV4** MUST either both be ANY or both be specified. The same applies to v6 tunnel endpoint fields.
* If the rule's **wFlags** field contains the FW\_CS\_RULE\_FLAGS\_DTM flag, then the rule is also a tunnel rule and the following requirements are relaxed: Either **dwRemoteTunnelEndpointV4** or **dwLocalTunnelEndpointV4**, or both, can now be empty. The same applies to the v6 tunnel endpoint fields. **Endpoint1** or **Endpoint2** or both can now be empty. The action can now also be FW\_CS\_RULE\_ACTION\_DO\_NOT\_SECURE.
* Tunnel endpoint addresses MUST NOT be the loopback addresses.
* If the **wFlags** field has the FW\_CS\_RULE\_FLAGS\_OUTBOUND\_CLEAR flag set or the FW\_CS\_RULE\_FLAGS\_TUNNEL\_BYPASS\_IF\_ENCRYPTED flag set, the rule MUST be a tunnel mode rule.

### FW\_CERT\_CRITERIA\_TYPE

The FW\_CERT\_CRITERIA\_TYPE enumeration defines whether the criteria are to be used for selection, validation, or both.

1. typedef enum
2. {
3. FW\_CERT\_CRITERIA\_TYPE\_BOTH,
4. FW\_CERT\_CRITERIA\_TYPE\_SELECTION,
5. FW\_CERT\_CRITERIA\_TYPE\_VALIDATION,
6. FW\_CERT\_CRITERIA\_TYPE\_MAX
7. } FW\_CERT\_CRITERIA\_TYPE;

**FW\_CERT\_CRITERIA\_TYPE\_BOTH:** Indicates that the criteria are to be used for both certificate selection and validation.

**FW\_CERT\_CRITERIA\_TYPE\_SELECTION:** Indicates that the criteria are meant for local certificate selection.

**FW\_CERT\_CRITERIA\_TYPE\_VALIDATION:** Indicates that the criteria are meant for validation of a peer certificate.

**FW\_CERT\_CRITERIA\_TYPE\_MAX:** To be valid, a value of this type MUST be less than this constant.

### FW\_CERT\_CRITERIA\_NAME\_TYPE

This enumeration defines the type of name to match in the certificate for a given criterion.

1. typedef enum
2. {
3. FW\_CERT\_CRITERIA\_NAME\_NONE,
4. FW\_CERT\_CRITERIA\_NAME\_DNS,
5. FW\_CERT\_CRITERIA\_NAME\_UPN,
6. FW\_CERT\_CRITERIA\_NAME\_RFC822,
7. FW\_CERT\_CRITERIA\_NAME\_CN,
8. FW\_CERT\_CRITERIA\_NAME\_OU,
9. FW\_CERT\_CRITERIA\_NAME\_O,
10. FW\_CERT\_CRITERIA\_NAME\_DC,
11. FW\_CERT\_CRITERIA\_NAME\_MAX
12. } FW\_CERT\_CRITERIA\_NAME\_TYPE;

**FW\_CERT\_CRITERIA\_NAME\_NONE:** Do not perform any name matching.

**FW\_CERT\_CRITERIA\_NAME\_DNS:** Match the DNS name in the Subject Alternative Name of the certificate.

**FW\_CERT\_CRITERIA\_NAME\_UPN:** Match the UPN name in the Subject Alternative Name of the certificate.

**FW\_CERT\_CRITERIA\_NAME\_RFC822:** Match the RFC822 name in the Subject Alternative Name of the certificate.

**FW\_CERT\_CRITERIA\_NAME\_CN:** Match the CN relative distinguished names (RDNs) in the Subject DN of the certificate.

**FW\_CERT\_CRITERIA\_NAME\_OU:** Match the OU RDNs in the Subject DN of the certificate.

**FW\_CERT\_CRITERIA\_NAME\_O:** Match the O RDNs in the Subject DN of the certificate.

**FW\_CERT\_CRITERIA\_NAME\_DC:** Match the DC RDNs in the Subject DN of the certificate.

**FW\_CERT\_CRITERIA\_NAME\_MAX:** To be valid, a value of this type MUST be less than this constant.

### FW\_CERT\_CRITERIA\_FLAGS

This enumeration describes bitmask flags that can be set on a criteria structure.

1. typedef enum
2. {
3. FW\_AUTH\_CERT\_CRITERIA\_FLAGS\_NONE = 0x0000,
4. FW\_AUTH\_CERT\_CRITERIA\_FLAGS\_FOLLOW\_RENEWAL = 0x0001,
5. FW\_AUTH\_CERT\_CRITERIA\_FLAGS\_MAX = 0x0002
6. } FW\_CERT\_CRITERIA\_FLAGS;

**FW\_AUTH\_CERT\_CRITERIA\_FLAGS\_NONE:** No flags are set.

**FW\_AUTH\_CERT\_CRITERIA\_FLAGS\_FOLLOW\_RENEWAL:** The renewal links in a certificate are to be followed, if they are found within a certificate.

F**W\_AUTH\_CERT\_CRITERIA\_FLAGS\_MAX:** To be valid, a flag value of this type MUST be less than this constant.

### FW\_CERT\_CRITERIA

This structure contains fields that are used when selecting a local certificate and validating a remote peer's certificate during certificate authentication.

1. typedef struct FW\_CERT\_CRITERIA {
2. WORD wSchemaVersion;
3. WORD wFlags;
4. FW\_CERT\_CRITERIA\_TYPE CertCriteriaType;
5. FW\_CERT\_CRITERIA\_NAME\_TYPE NameType;
6. LPWSTR wszName;
7. DWORD dwNumEku;
8. LPSTR ppEku;
9. LPWSTR wszHash;
10. } FW\_CERT\_CRITERIA,
11. \*PFW\_CERT\_CRITERIA;

**wSchemaVersion:**  Specifies the version of the criteria structure.

**wFlags:**  A WORD containing bit flags, whose value is defined in [FW\_CERT\_CRITERIA\_FLAGS](#Section_df8aba33578c4f01802c435306e63409). The flag FW\_AUTH\_CERT\_CRITERIA\_FLAGS\_FOLLOW\_RENEWAL MUST NOT be set if the field **wszHash** is null. If specified, the flag FW\_AUTH\_CERT\_CRITERIA\_FLAGS\_FOLLOW\_RENEWAL MUST NOT be used if **CertCriteriaType** is equal to FW\_CERT\_CRITERIA\_TYPE\_VALIDATION.

**CertCriteriaType:**  Specifies the type of criteria used, as among those specified in the [FW\_CERT\_CRITERIA\_TYPE](#Section_84a1305fff3a4bbeba81e1c111ff5ba4) enumeration. This value MUST be less than FW\_CERT\_CRITERIA\_TYPE\_MAX.

**NameType:**  Specifies the type of name, as among those specified in the [FW\_CERT\_CRITERIA\_NAME\_TYPE](#Section_b83533eb9e4142558dd4be8bf75d1053) enumeration. This value MUST be less than FW\_CERT\_CRITERIA\_NAME\_MAX. If the value is not equal to FW\_CERT\_CRITERIA\_NAME\_NONE, then the value for wszName MUST be specified.

**wszName:**  A [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string that specifies a name corresponding to the **NameType** specified. The length of this Unicode string MUST be less than 10,000 characters. The name MUST not contain the pipe "|" character.

**dwNumEku:**  Specifies the number of [**EKU**](#gt_06beeb29-6e93-4472-a53d-bbd51eca5759) element entries in the **ppEku** array.

**ppEku:**  Pointer to an array of pointers to null-terminated strings. Each string in the array MUST contain only characters in the range "0" to "9" or the "." character. The number of elements in the array MUST be equal to the value of the **dwNumEku** field.

**wszHash:**  A Unicode string that specifies the hash of the certificate. The number of characters in the string MUST be equal to 40. Each character in the string MUST be in one of the following ranges: "0" to "9", "a" to "f", or "A" to "F".

### FW\_AUTH\_METHOD

This enumeration defines the different authentication methods that are used for authentication. The **IpSecPhase** field of the [FW\_AUTH\_SET](#Section_bbec7ab70a1d49a49c8ea4784bbe6f16) containing the [FW\_AUTH\_SUITE](#Section_447c3a4c9543495dab7db7ca639712ef) determines which authentication methods are valid for a particular authentication suite.

1. typedef enum \_tag\_FW\_AUTH\_METHOD
2. {
3. FW\_AUTH\_METHOD\_INVALID = 0,
4. FW\_AUTH\_METHOD\_ANONYMOUS = 1,
5. FW\_AUTH\_METHOD\_MACHINE\_KERB = 2,
6. FW\_AUTH\_METHOD\_MACHINE\_SHKEY = 3,
7. FW\_AUTH\_METHOD\_MACHINE\_NTLM = 4,
8. FW\_AUTH\_METHOD\_MACHINE\_CERT = 5,
9. FW\_AUTH\_METHOD\_USER\_KERB = 6,
10. FW\_AUTH\_METHOD\_USER\_CERT = 7,
11. FW\_AUTH\_METHOD\_USER\_NTLM = 8,
12. FW\_AUTH\_METHOD\_MACHINE\_RESERVED = 9,
13. FW\_AUTH\_METHOD\_USER\_RESERVED = 10,
14. FW\_AUTH\_METHOD\_MAX\_2\_10 = 9,
15. FW\_AUTH\_METHOD\_MAX = 11
16. } FW\_AUTH\_METHOD;

**FW\_AUTH\_METHOD\_INVALID:** This value MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 0.

**FW\_AUTH\_METHOD\_ANONYMOUS:** This method does not require identity to authenticate. It is equal to no authentication. This method can be used for both FW\_IPSEC\_PHASE\_1 or FW\_IPSEC\_PHASE\_2. This symbolic constant has a value of 1.

**FW\_AUTH\_METHOD\_MACHINE\_KERB:** This method authenticates the identity of machines by using Kerberos Protocol Extensions as specified in [[MS-KILE]](%5bMS-KILE%5d.pdf#Section_2a32282edd484ad9a542609804b02cc9). This method MUST be used only on FW\_IPSEC\_PHASE\_1. This symbolic constant has a value of 2.

**FW\_AUTH\_METHOD\_MACHINE\_SHKEY:** This method uses a previous manually shared key to authenticate machine identities. This method MUST be used only on FW\_IPSEC\_PHASE\_1. This symbolic constant has a value of 3.

**FW\_AUTH\_METHOD\_MACHINE\_NTLM:** This method authenticates the identity of machines by using the NTLM Authentication Protocol as specified in [[MS-NLMP]](%5bMS-NLMP%5d.pdf#Section_b38c36ed28044868a9ff8dd3182128e4). This method MUST be used only on FW\_IPSEC\_PHASE\_1. This symbolic constant has a value of 4.

**FW\_AUTH\_METHOD\_MACHINE\_CERT:** This method authenticates the identity of a machine by using machine certificates. This method can be used for both FW\_IPSEC\_PHASE\_1 or FW\_IPSEC\_PHASE\_2. This symbolic constant has a value of 5.

**FW\_AUTH\_METHOD\_USER\_KERB:** This method authenticates user identities by using the Kerberos Protocol Extensions. This method MUST be used only on FW\_IPSEC\_PHASE\_2. This symbolic constant has a value of 6.

**FW\_AUTH\_METHOD\_USER\_CERT:** This method authenticates user identities by using user certificates. This method MUST be used only on FW\_IPSEC\_PHASE\_2. This symbolic constant has a value of 7.

**FW\_AUTH\_METHOD\_USER\_NTLM:** This method authenticates user identities by using the NTLM Authentication Protocol. This method MUST be used only on FW\_IPSEC\_PHASE\_2. This symbolic constant has a value of 8.

**FW\_AUTH\_METHOD\_MACHINE\_RESERVED:** This value is invalid and MUST NOT be used. This symbolic constant has a value of 9.

**FW\_AUTH\_METHOD\_USER\_RESERVED:** This value is invalid and MUST NOT be used. This symbolic constant has a value of 10.

**FW\_AUTH\_METHOD\_MAX\_2\_10:** This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x020A and earlier. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 9.

**FW\_AUTH\_METHOD\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 11.

### FW\_AUTH\_SUITE\_FLAGS

This enumeration describes bitmask flags that can be set on authentication proposals.

1. typedef enum \_tag\_FW\_AUTH\_SUITE\_FLAGS
2. {
3. FW\_AUTH\_SUITE\_FLAGS\_NONE = 0x0000,
4. FW\_AUTH\_SUITE\_FLAGS\_CERT\_EXCLUDE\_CA\_NAME = 0x0001,
5. FW\_AUTH\_SUITE\_FLAGS\_HEALTH\_CERT = 0x0002,
6. FW\_AUTH\_SUITE\_FLAGS\_PERFORM\_CERT\_ACCOUNT\_MAPPING = 0x0004,
7. FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA256 = 0x0008,
8. FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA384 = 0x0010,
9. FW\_AUTH\_SUITE\_FLAGS\_INTERMEDIATE\_CA = 0x0020,
10. W\_AUTH\_SUITE\_FLAGS\_ALLOW\_PROXY = 0x0040,
11. FW\_AUTH\_SUITE\_FLAGS\_MAX = 0x0080,
12. FW\_AUTH\_SUITE\_FLAGS\_MAX\_V2\_1 = 0x0020
13. } FW\_AUTH\_SUITE\_FLAGS;

**FW\_AUTH\_SUITE\_FLAGS\_NONE:** This value means that none of the following flags are set. This value is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code.

**FW\_AUTH\_SUITE\_FLAGS\_CERT\_EXCLUDE\_CA\_NAME:** If this flag is set, [**certificate authority (CA)**](#gt_c925d5d7-a442-4ba4-9586-5f94ccec847a) names are excluded. This flag MUST be set only on first authentications.

**FW\_AUTH\_SUITE\_FLAGS\_HEALTH\_CERT:** This flag specifies that the certificate in use is a health certificate. On second authentications, if the authentication method is using a machine certificate, this flag MUST be specified. Also on second authentications, if the authentication method is using a user certificate, this flag MUST NOT be specified.

**FW\_AUTH\_SUITE\_FLAGS\_PERFORM\_CERT\_ACCOUNT\_MAPPING:** This flag specifies that the certificate that is used maps to an account.

**FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA256:** This flag specifies that the default certificate signing algorithm of [**RSA**](#gt_3f85a24a-f32a-4322-9e99-eba6ae802cd6) MUST be replaced by the Elliptic Curve Digital Signature Algorithm (ECDSA) using curves with a 256-bit prime moduli.

**FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA384:** This flag specifies that the default certificate signing algorithm of RSA MUST be replaced by the Elliptic Curve Digital Signature Algorithm using curves with a 384-bit prime moduli.

**FW\_AUTH\_SUITE\_FLAGS\_INTERMEDIATE\_CA:** This flag specifies that the certificate used is not from a root certificate authority but from an intermediate authority in the chain.

**W\_AUTH\_SUITE\_FLAGS\_ALLOW\_PROXY:** This flag specifies that the host machine MUST use a proxy server to communicate with the [**Key Distribution Center (KDC)**](#gt_6e5aafba-6b66-4fdd-872e-844f142af287) when performing [**Kerberos**](#gt_d6a282ce-b1da-41e1-b05a-22f777a5c1fe) authentication. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used.

**FW\_AUTH\_SUITE\_FLAGS\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 0x0080.

**FW\_AUTH\_SUITE\_FLAGS\_MAX\_V2\_1:** This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x0201 and earlier. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 0x0020.

### FW\_AUTH\_SUITE2\_10

This structure describes an [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) authentication suite. An authentication suite is a proposal of a set of algorithms and parameters that specify the authentication method to be used. It also includes some modifiers and parameters for the authentication method.

1. typedef struct \_tag\_FW\_AUTH\_SUITE2\_10 {
2. [range(FW\_AUTH\_METHOD\_INVALID+1, FW\_AUTH\_METHOD\_MAX)]
3. FW\_AUTH\_METHOD Method;
4. unsigned short wFlags;
5. [switch\_type(FW\_AUTH\_METHOD), switch\_is(Method)]
6. union {
7. [case(FW\_AUTH\_METHOD\_MACHINE\_CERT,FW\_AUTH\_METHOD\_USER\_CERT)]
8. struct {
9. [ref, string] wchar\_t\* wszCAName;
10. };
11. [case(FW\_AUTH\_METHOD\_MACHINE\_SHKEY)]
12. struct {
13. [ref, string] wchar\_t\* wszSHKey;
14. };
15. [default]  ;
16. };
17. } FW\_AUTH\_SUITE2\_10,
18. \*PFW\_AUTH\_SUITE2\_10;

**Method:**  This field is of type [FW\_AUTH\_METHOD](#Section_59e71d46440747d29cf58889fd3a74f2). It specifies the authentication method that is suggested by this proposal suite.

**wFlags:**  This flag is a combination of flags from [FW\_AUTH\_SUITE\_FLAGS](#Section_ff03b48b87cf4020bf0b7f4c680d9b02).

**wszCAName:**  A pointer to a [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string. This string represents the name of the [**certificate authority**](#gt_c925d5d7-a442-4ba4-9586-5f94ccec847a) to be used to authenticate when using machine or user certificate methods.

**wszSHKey:**  A pointer to a Unicode string. This string is the previous, manually shared secret that is used to authenticate when using preshared key methods.

If the method is machine certificate or user certificate, the **wszCAName** string MUST NOT be NULL, MUST be at least 1 character long, MUST NOT be greater than or equal to 10,000 characters, MUST NOT contain the pipe(|) character, and MUST be a CERT\_X500\_NAME\_STR string type name encoded with X509\_ASN\_ENCODING. If the method is SHKEY, the **wszSHKey** string MUST NOT be NULL, MUST be at least 1 character long, MUST NOT be greater than or equal to 10,000 characters, and MUST NOT contain the pipe (|) character.

### FW\_AUTH\_SUITE

This structure specifies an [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) authentication suite and includes certification selection criteria. An authentication suite is a proposal of a set of algorithms and parameters that specify the authentication method to be used.

1. typedef struct \_tag\_FW\_AUTH\_SUITE {
2. [range(FW\_AUTH\_METHOD\_INVALID+1, FW\_AUTH\_METHOD\_MAX)]
3. FW\_AUTH\_METHOD Method;
4. unsigned short wFlags;
5. [switch\_type(FW\_AUTH\_METHOD), switch\_is(Method)]
6. union {
7. [case(FW\_AUTH\_METHOD\_MACHINE\_CERT,FW\_AUTH\_METHOD\_USER\_CERT)]
8. struct {
9. [ref, string] wchar\_t\* wszCAName;
10. [unique] PFW\_CERT\_CRITERIA pCertCriteria;
11. };
12. [case(FW\_AUTH\_METHOD\_MACHINE\_SHKEY)]
13. struct {
14. [ref, string] wchar\_t\* wszSHKey;
15. } pCertCriteria;
16. [case(FW\_AUTH\_METHOD\_MACHINE\_KERB, FW\_AUTH\_METHOD\_USER\_KERB)]
17. struct {
18. [unique, string] WCHAR\* wszProxyServer;
19. };
20. [default]  ;
21. };
22. } FW\_AUTH\_SUITE,
23. \*PFW\_AUTH\_SUITE;

**Method:**  This field is of type [FW\_AUTH\_METHOD](#Section_59e71d46440747d29cf58889fd3a74f2). It specifies the authentication method that is suggested by this proposal suite.

**wFlags:**  This flag is a combination of flags from [FW\_AUTH\_SUITE\_FLAGS](#Section_ff03b48b87cf4020bf0b7f4c680d9b02).

**wszCAName:**  A pointer to a [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string. This string represents the name of the [**certificate authority**](#gt_c925d5d7-a442-4ba4-9586-5f94ccec847a) to be used to authenticate when using machine or user certificate methods.

**pCertCriteria:**  A pointer to a structure of type [PFW\_CERT\_CRITERIA](#Section_8f4e2f51826f43c794055395b824bd41). This field MUST NOT be present unless the **Method** field has the value FW\_AUTH\_METHOD\_MACHINE\_CERT or FW\_AUTH\_METHOD\_USER\_CERT.

It contains fields which are used when selecting a local certificate and validating a remote peer's certificate during certificate authentication.

**wszSHKey:**  A pointer to a Unicode string. This string is the previous, manually shared secret that is used to authenticate when using preshared key methods.

**wszProxyServer:**  A pointer to a Unicode string specifying the [**fully qualified domain name (FQDN)**](#gt_1769aec9-237e-44ed-9014-1abb3ec6de6e) of the [**Kerberos**](#gt_d6a282ce-b1da-41e1-b05a-22f777a5c1fe) proxy server. This field MUST be set if and only if the FW\_AUTH\_SUITE\_FLAGS\_ALLOW\_PROXY flag is set.

If the method is machine certificate or user certificate, the **wszCAName** string MUST NOT be NULL, MUST be at least 1 character long, MUST NOT be greater than or equal to 10,000 characters, MUST NOT contain the pipe(|) character, and MUST be a valid Name as defined in [[X501]](https://go.microsoft.com/fwlink/?LinkId=98847) section 9.2. If the method is SHKEY, the **wszSHKey** string MUST NOT be NULL, MUST be at least 1 character long, MUST NOT be greater than or equal to 10,000 characters, and MUST NOT contain the pipe (|) character.

If the **Method** is not FW\_AUTH\_METHOD\_MACHINE\_CERT or FW\_AUTH\_METHOD\_USER\_CERT then the **pCertCriteria** field MUST be NULL.

### FW\_AUTH\_SET2\_10

This structure contains a list of [FW\_AUTH\_SUITE2\_10](#Section_bae6ee78bedc4a3a95458920c2a93cda) elements that are ordered from highest to lowest preference and are negotiated with remote peers to establish authentication algorithms.

1. typedef struct \_tag\_FW\_AUTH\_SET2\_10 {
2. struct \_tag\_FW\_AUTH\_SET2\_10\* pNext;
3. unsigned short wSchemaVersion;
4. [range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)]
5. FW\_IPSEC\_PHASE IpSecPhase;
6. [string, range(1,255), ref] wchar\_t\* wszSetId;
7. [string, range(1,10001)] wchar\_t\* wszName;
8. [string, range(1,10001)] wchar\_t\* wszDescription;
9. [string, range(1,10001)] wchar\_t\* wszEmbeddedContext;
10. [range(0,1000)] unsigned long dwNumSuites;
11. [size\_is(dwNumSuites)] PFW\_AUTH\_SUITE pSuites;
12. [range(FW\_RULE\_ORIGIN\_INVALID,FW\_RULE\_ORIGIN\_MAX-1)]
13. FW\_RULE\_ORIGIN\_TYPE Origin;
14. [string, range(1,10001)] wchar\_t\* wszGPOName;
15. FW\_RULE\_STATUS Status;
16. unsigned long dwAuthSetFlags;
17. } FW\_AUTH\_SET2\_10,
18. \*PFW\_AUTH\_SET2\_10;

**pNext:**  A pointer to the next FW\_AUTH\_SET2\_10 in the list.

**wSchemaVersion:**  Specifies the version of the set.

**IpSecPhase:**  This field is of type [FW\_IPSEC\_PHASE](#Section_e23d75abc5f14b77b290f0e0e25d72e1), and it specifies if this authentication set applies for first or second authentications.

**wszSetId:**  A pointer to a [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string that uniquely identifies the set. The default set for this policy object is identified with the "{E5A5D32A-4BCE-4e4d-B07F-4AB1BA7E5FE3}" string for Phase1 and the "{E5A5D32A-4BCE-4e4d-B07F-4AB1BA7E5FE4}" string for Phase2. Default sets are merged across policy stores, and only one is enforced according to predefined merge logic rules.

**wszName:**  A pointer to a Unicode string that provides a friendly name for the set.

**wszDescription:**  A pointer to a Unicode string that provides a friendly description for the set.

**wszEmbeddedContext:**  A pointer to a Unicode string that provides a way for applications to store relevant application-specific context that is related to the set.

**dwNumSuites:**  Specifies the number of authentication suites that the structure contains.

**pSuites:**  A pointer to an array of [FW\_AUTH\_SUITE](#Section_447c3a4c9543495dab7db7ca639712ef) elements. The number of elements is given by **dwNumSuites**.

**Origin:**  This field is the set origin, as specified in the [FW\_RULE\_ORIGIN\_TYPE](#Section_9d295321d75c41c0ab0d7a78df40f77c) enumeration. It MUST be filled on enumerated rules and ignored on input.

**wszGPOName:**  A Unicode string that represents the name of the originating [**GPO**](#gt_dec32233-8776-4151-91a0-8624a2b9abb0). It MUST be set if the origin is [**Group Policy**](#gt_defe8c22-1365-4e5e-abf7-46ad112d3bda); otherwise, it MUST be NULL.

**Status:**  A status code of the set, as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out when the structure is returned as output. On input, this field MUST be set to FW\_RULE\_STATUS\_OK.

**dwAuthSetFlags:**  A reserved value and not currently used. It MUST be set to 0.

The following are semantic checks that authentication sets MUST pass:

* The **wSchemaVersion** field MUST NOT be less than 0x000200.
* The **wszSetId** field MUST NOT contain the pipe (|) character, MUST NOT be NULL, MUST be a string of at least 1 character long, and MUST NOT be greater than or equal to 255 characters.
* If the **wszName** field string is not NULL, it MUST be at least 1 character long, MUST NOT be greater than or equal to 10,000 characters, and MUST NOT contain the pipe (|) character.
* If the **wszDescription** field string is not NULL, it MUST be at least 1 character long, MUST NOT be greater than or equal to 10,000 characters, and MUST NOT contain the pipe (|) character.
* If the **wszEmbeddedContext** field string is not NULL, it MUST be at least 1 character long, MUST NOT be greater than or equal to 10,000 characters, and MUST NOT contain the pipe (|) character.
* The **IpSecPhase** field MUST have valid FW\_IPSEC\_PHASE values.
* If **IpSecPhase** is FW\_IPSEC\_PHASE\_1:
  + The **wszSetId** field MUST NOT have the default phase 1 authentication set ID as a prefix.
  + The authentication set MUST have at least one authentication suite.
  + The **dwNumSuites** field MUST agree with the **pSuites** field.
  + The authentication suites methods MUST only be FW\_AUTH\_METHOD\_ANONYMOUS, FW\_AUTH\_METHOD\_MACHINE\_KERB, FW\_AUTH\_METHOD\_MACHINE\_NTLM, FW\_AUTH\_METHOD\_MACHINE\_CERT, or FW\_AUTH\_METHOD\_MACHINE\_SHKEY.
  + Authentication suites that have a method other than machine certificate MUST have the **wFlags** field of the same suite set to 0.
  + If the set schema policy version is 0x200, the **wFlags** field MUST NOT contain the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA256 or the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA384 flags.
  + The **wFlags** field MUST NOT contain both the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA256 and the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA384 flags.
  + All suites that have the FW\_AUTH\_METHOD\_MACHINE\_CERT method and a **wFlags** field with the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA256 flag set, MUST be contiguous. The same applies for those suites that have the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA384 flag set, and those suites that have neither flag set (they default to [**RSA**](#gt_3f85a24a-f32a-4322-9e99-eba6ae802cd6) signing).
  + All such contiguous suites that have a specific signing flag (either none, ECDSA256, or ECDSA384) MUST have the same value for the FW\_AUTH\_SUITE\_FLAG\_HEALTH\_CERT flag. It MUST be set either in all or in none.
  + The set MUST NOT have more than one suite that has the anonymous method (FW\_AUTH\_METHOD\_ANONYMOUS), or that has the machine kerb method (FW\_AUTH\_METHOD\_MACHINE\_KERB), or that has the machine ntlm method (FW\_AUTH\_METHOD\_MACHINE\_NTLM), or that has the machine shkey method (FW\_AUTH\_METHOD\_MACHINE\_SHKEY), as defined in section [2.2.59](#Section_59e71d46440747d29cf58889fd3a74f2).[<14>](#Appendix_A_14" \o "Product behavior note 14)
  + The set MUST NOT have a suite that has an NTLM Authentication Protocol method (as specified in [[MS-NLMP]](%5bMS-NLMP%5d.pdf#Section_b38c36ed28044868a9ff8dd3182128e4)) and a suite SHKey method.
  + If the set has a machine certificate suite that has a **wFlag** that contains the flag FW\_AUTH\_SUITE\_FLAGS\_HEALTH\_CERT, all machine certificate method suites in the set MUST also have this flag.
  + If the set schema policy version is less than 0x214, the set MUST NOT have suites that contain the FW\_AUTH\_METHOD\_MACHINE\_NEGOEX authentication method.
* If the **IpSecPhase** is FW\_IPSEC\_PHASE\_2:
  + The **wszSetId** MUST NOT have the default phase 2 authentication set ID as a prefix.
  + The **dwNumSuites** field MUST agree with the **pSuites** field.
  + The authentication suites methods MUST only be FW\_AUTH\_METHOD\_ANONYMOUS, FW\_AUTH\_METHOD\_USER\_KERB, FW\_AUTH\_METHOD\_USER\_NTLM, FW\_AUTH\_METHOD\_USER\_CERT, or FW\_AUTH\_METHOD\_MACHINE\_CERT.
  + The set MUST NOT have a suite that has the anonymous method as the only suite.
  + Suites in the set MUST NOT contain FW\_AUTH\_SUITE\_FLAGS\_CERT\_EXCLUDE\_CA\_NAME.
  + Suites that have user certificate methods MUST NOT contain the FW\_AUTH\_SUITE\_FLAGS\_HEALTH\_CERT flag; however, suites that have machine certificate methods MUST contain it.
  + Authentication suites that have a method other than machine certificate or user certificate MUST have the **wFlags** field of the same suite set to 0.
  + If the set schema policy version is 0x200, the **wFlags** field MUST NOT contain the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA256 or the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA384 flags.
  + The **wFlags** field MUST NOT contain both the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA256 and the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA384 flags.
  + All suites that have a FW\_AUTH\_METHOD\_MACHINE\_CERT method and a **wFlags** field with the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA256 flag set, MUST be contiguous. The same applies to those suites that have the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA384 flag set and those suites that have neither flag set (they default to RSA signing).
  + The set MUST NOT have more than one suite that has the anonymous method (FW\_AUTH\_METHOD\_ANONYMOUS), or that has the user kerb method (FW\_AUTH\_METHOD\_USER\_KERB), or that has the user ntlm method (FW\_AUTH\_METHOD\_USER\_NTLM), as defined in section 2.2.59.[<15>](#Appendix_A_15" \o "Product behavior note 15)
  + A set that contains a suite that has the machine certificate method MUST NOT contain suites that have the user certificate method.
  + A set that contains a suite that has the machine certificate method MUST only contain more suites that have machine certificate or anonymous methods.
  + If the set schema policy version is less than 0x214, the set MUST NOT have suites that contain the FW\_AUTH\_METHOD\_USER\_NEGOEX authentication method.

### FW\_AUTH\_SET

This structure contains a list of [FW\_AUTH\_SUITE](#Section_447c3a4c9543495dab7db7ca639712ef) elements that are ordered from highest to lowest preference and are negotiated with remote peers to establish authentication algorithms.

1. typedef struct \_tag\_FW\_AUTH\_SET {
2. struct \_tag\_FW\_AUTH\_SET\* pNext;
3. unsigned short wSchemaVersion;
4. [range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)]
5. FW\_IPSEC\_PHASE IpSecPhase;
6. [string, range(1,255), ref] wchar\_t\* wszSetId;
7. [string, range(1,10001)] wchar\_t\* wszName;
8. [string, range(1,10001)] wchar\_t\* wszDescription;
9. [string, range(1,10001)] wchar\_t\* wszEmbeddedContext;
10. [range(0,1000)] unsigned long dwNumSuites;
11. [size\_is(dwNumSuites)] PFW\_AUTH\_SUITE pSuites;
12. [range(FW\_RULE\_ORIGIN\_INVALID,FW\_RULE\_ORIGIN\_MAX-1)]
13. FW\_RULE\_ORIGIN\_TYPE Origin;
14. [string, range(1,10001)] wchar\_t\* wszGPOName;
15. FW\_RULE\_STATUS Status;
16. unsigned long dwAuthSetFlags;
17. } FW\_AUTH\_SET,
18. \*PFW\_AUTH\_SET;

**pNext:**  A pointer to the next FW\_AUTH\_SET in the list.

**wSchemaVersion:**  Specifies the version of the set.

**IpSecPhase:**  This field is of type [FW\_IPSEC\_PHASE](#Section_e23d75abc5f14b77b290f0e0e25d72e1), and it specifies if this authentication set applies for first or second authentications.

**wszSetId:**  A pointer to a [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string that uniquely identifies the set. The primary set for this policy object is identified with the "{E5A5D32A-4BCE-4e4d-B07F-4AB1BA7E5FE3}" string for Phase1 and the "{E5A5D32A-4BCE-4e4d-B07F-4AB1BA7E5FE4}" string for Phase2.

**wszName:**  A pointer to a Unicode string that provides a friendly name for the set.

**wszDescription:**  A pointer to a Unicode string that provides a friendly description for the set.

**wszEmbeddedContext:**  A pointer to a Unicode string that provides a way for applications to store relevant application-specific context that is related to the set.

**dwNumSuites:**  Specifies the number of authentication suites that the structure contains.

**pSuites:**  A pointer to an array of FW\_AUTH\_SUITE elements. The number of elements is given by **dwNumSuites**.

**Origin:**  This field is the set origin, as specified in the [FW\_RULE\_ORIGIN\_TYPE](#Section_9d295321d75c41c0ab0d7a78df40f77c) enumeration. It MUST be filled on enumerated rules and ignored on input.

**wszGPOName:**  A pointer to a Unicode string containing the displayName of the [**GPO**](#gt_dec32233-8776-4151-91a0-8624a2b9abb0) containing this object. When adding a new object, this field is not used. The client SHOULD set the value to NULL, and the server MUST ignore the value. When enumerating an existing object, if the client does not set the FW\_ENUM\_RULES\_FLAG\_RESOLVE\_GPO\_NAME flag, the server MUST set the value to NULL. Otherwise, the server MUST set the value to the displayName of the GPO containing the object or NULL if the object is not contained within a GPO. For details about how the server initializes an object from a GPO, see section [3.1.3](#Section_e8924ac5aa4a41d1bf654f46b3d399aa). For details about how the displayName of a GPO is stored, see [[MS-GPOL]](%5bMS-GPOL%5d.pdf#Section_62d1292462524052996f161d2b9019f4) section 2.3.

**Status:**  The status code of the set which MUST be one of the values defined in the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field's value is assigned when the structure is returned as output. When first sent, this field MUST be set to FW\_RULE\_STATUS\_OK.

**dwAuthSetFlags:**  Bit flags from FW\_AUTH\_SET\_FLAGS.

The following are semantic checks that authentication sets MUST pass:

* The **wSchemaVersion** field MUST NOT be less than 0x000200.
* The **wszSetId** field MUST NOT contain the pipe (|) character, MUST NOT be NULL, MUST be a string of at least 1 character long, and MUST NOT be greater than or equal to 255 characters.
* If the **wszName** field string is not NULL, it MUST be at least 1 character long, MUST NOT be greater than or equal to 10,000 characters, and MUST NOT contain the pipe (|) character.
* If the **wszDescription** field string is not NULL, it MUST be at least 1 character long, MUST NOT be greater than or equal to 10,000 characters, and MUST NOT contain the pipe (|) character.
* If the **wszEmbeddedContext** field string is not NULL, it MUST be at least 1 character long, its length MUST NOT be greater than or equal to 10,000 characters, and MUST NOT contain the pipe (|) character.

If the method of a suite is machine certificate or user certificate, and its **pCertCriteria** field is not NULL, then the **wSchemaVersion** of the **pCertCriteria** field MUST be equal to the schema version specified in the **wSchemaVersion** field of the auth set containing the suite.

* The **IpSecPhase** field MUST have valid FW\_IPSEC\_PHASE values.
* If **IpSecPhase** is FW\_IPSEC\_PHASE\_1:
  + The **wszSetId** field MUST NOT have the primary phase 1 authentication set ID as a prefix.
  + The authentication set MUST have at least one authentication suite.
  + The **dwNumSuites** field MUST agree with the **pSuites** field.
  + The authentication suites methods MUST each be either FW\_AUTH\_METHOD\_ANONYMOUS, FW\_AUTH\_METHOD\_MACHINE\_KERB, FW\_AUTH\_METHOD\_MACHINE\_NTLM, FW\_AUTH\_METHOD\_MACHINE\_CERT, or FW\_AUTH\_METHOD\_MACHINE\_SHKEY.
  + Authentication suites that have a method other than machine certificate MUST have the **wFlags** field of the same suite set to 0.
  + If the set schema policy version is 0x200, the **wFlags** field MUST NOT contain the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA256 or the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA384 flags.
  + The **wFlags** field MUST NOT contain both the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA256 and the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA384 flags.
  + All suites that have the FW\_AUTH\_METHOD\_MACHINE\_CERT method and a **wFlags** field with the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA256 flag set, MUST be contiguous. The same applies for those suites that have the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA384 flag set, and those suites that have neither flag set (they default to [**RSA**](#gt_3f85a24a-f32a-4322-9e99-eba6ae802cd6) signing).
  + All such contiguous suites that have a specific signing flag (either none, ECDSA256, or ECDSA384) MUST have the same value for the FW\_AUTH\_SUITE\_FLAG\_HEALTH\_CERT flag.
  + The set MUST NOT have more than one suite that has the anonymous method (FW\_AUTH\_METHOD\_ANONYMOUS), or that has the machine kerb method (FW\_AUTH\_METHOD\_MACHINE\_KERB), or that has the machine ntlm method (FW\_AUTH\_METHOD\_MACHINE\_NTLM), or that has the machine shkey method (FW\_AUTH\_METHOD\_MACHINE\_SHKEY), as defined in section [2.2.59](#Section_59e71d46440747d29cf58889fd3a74f2).[<16>](#Appendix_A_16" \o "Product behavior note 16)
  + The set MUST NOT have a suite that has an NTLM Authentication Protocol method (as specified in [[MS-NLMP]](%5bMS-NLMP%5d.pdf#Section_b38c36ed28044868a9ff8dd3182128e4)) and a suite SHKey method.
  + If the set has a machine certificate suite that has a **wFlag** that contains the flag FW\_AUTH\_SUITE\_FLAGS\_HEALTH\_CERT, all machine certificate method suites in the set MUST also have this flag.
  + If the set schema policy version is less than 0x214, the set MUST NOT have suites that contain the FW\_AUTH\_METHOD\_MACHINE\_NEGOEX authentication method.
* If the **IpSecPhase** is FW\_IPSEC\_PHASE\_2:
  + The **wszSetId** MUST NOT have the primary phase 2 authentication set ID as a prefix.
  + The **dwNumSuites** field MUST agree with the **pSuites** field.
  + The authentication suites methods MUST each be one of FW\_AUTH\_METHOD\_ANONYMOUS, FW\_AUTH\_METHOD\_USER\_KERB, FW\_AUTH\_METHOD\_USER\_NTLM, FW\_AUTH\_METHOD\_USER\_CERT, or FW\_AUTH\_METHOD\_MACHINE\_CERT.
  + The set MUST NOT have a suite that has the anonymous method as the only suite.
  + Suites in the set MUST NOT contain FW\_AUTH\_SUITE\_FLAGS\_CERT\_EXCLUDE\_CA\_NAME.
  + Suites that have user certificate methods MUST NOT contain the FW\_AUTH\_SUITE\_FLAGS\_HEALTH\_CERT flag; however, suites that have machine certificate methods MUST contain it.
  + Authentication suites that have a method other than machine certificate or user certificate MUST have the **wFlags** field of the same suite set to 0.
  + If the set schema policy version is 0x200, the **wFlags** field MUST NOT contain the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA256 or the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA384 flags.
  + The **wFlags** field MUST NOT contain both the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA256 and the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA384 flags.
  + All suites that have a FW\_AUTH\_METHOD\_MACHINE\_CERT method and a **wFlags** field with the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA256 flag set, MUST be contiguous. The same applies to those suites that have the FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA384 flag set and those suites that have neither flag set (they default to RSA signing).
  + The set MUST NOT have more than one suite that has the anonymous method (FW\_AUTH\_METHOD\_ANONYMOUS), or that has the user kerb method (FW\_AUTH\_METHOD\_USER\_KERB), or that has the user ntlm method (FW\_AUTH\_METHOD\_USER\_NTLM), as defined in section 2.2.59.[<17>](#Appendix_A_17" \o "Product behavior note 17)
  + A set that contains a suite that has the machine certificate method MUST NOT contain suites that have the user certificate method.
  + A set that contains a suite that has the machine certificate method MUST only contain suites that have machine certificate or anonymous methods.

### FW\_CRYPTO\_KEY\_EXCHANGE\_TYPE

This enumeration is used to identify supported key exchange algorithms.

1. typedef enum \_tag\_FW\_CRYPTO\_KEY\_EXCHANGE\_TYPE
2. {
3. FW\_CRYPTO\_KEY\_EXCHANGE\_NONE = 0,
4. FW\_CRYPTO\_KEY\_EXCHANGE\_DH1 = 1,
5. FW\_CRYPTO\_KEY\_EXCHANGE\_DH2 = 2,
6. FW\_CRYPTO\_KEY\_EXCHANGE\_ECDH256 = 3,
7. FW\_CRYPTO\_KEY\_EXCHANGE\_ECDH384 = 4,
8. FW\_CRYPTO\_KEY\_EXCHANGE\_DH14 = 5,
9. FW\_CRYPTO\_KEY\_EXCHANGE\_DH14 = FW\_CRYPTO\_KEY\_EXCHANGE\_DH2048 = 5,
10. FW\_CRYPTO\_KEY\_EXCHANGE\_DH24 = 6,
11. FW\_CRYPTO\_KEY\_EXCHANGE\_MAX\_V2\_10 = FW\_CRYPTO\_KEY\_EXCHANGE\_DH24 = 6,
12. FW\_CRYPTO\_KEY\_EXCHANGE\_MAX = 7
13. } FW\_CRYPTO\_KEY\_EXCHANGE\_TYPE;

**FW\_CRYPTO\_KEY\_EXCHANGE\_NONE:** This value means that there are no key exchange algorithms defined. When enumerating [**SAs**](#gt_67cbf867-7a49-41f3-a68f-37b5f9035acb), this value MAY be returned. It MUST NOT be used for other cases. This symbolic constant has a value of 0.

**FW\_CRYPTO\_KEY\_EXCHANGE\_DH1:** Do key exchange with Diffie-Hellman group 1. This symbolic constant has a value of 1.

**FW\_CRYPTO\_KEY\_EXCHANGE\_DH2:** Do key exchange with Diffie-Hellman group 2. This symbolic constant has a value of 2.

**FW\_CRYPTO\_KEY\_EXCHANGE\_ECDH256:** Do key exchange with elliptic curve Diffie-Hellman 256. This symbolic constant has a value of 3.

**FW\_CRYPTO\_KEY\_EXCHANGE\_ECDH384:** Do key exchange with elliptic curve Diffie-Hellman 384. This symbolic constant has a value of 4.

**FW\_CRYPTO\_KEY\_EXCHANGE\_DH14:** Do key exchange with Diffie-Hellman group 14. This symbolic constant has a value of 5.

**FW\_CRYPTO\_KEY\_EXCHANGE\_DH14 = FW\_CRYPTO\_KEY\_EXCHANGE\_DH2048:** Do key exchange with Diffie-Hellman group 14. This group was called Diffie-Hellman group 2048 when it was introduced. The name has been changed to match standard terminology. This symbolic constant has a value of 5.

**FW\_CRYPTO\_KEY\_EXCHANGE\_DH24:** Do key exchange with Diffie-Hellman group 24. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used. This symbolic constant has a value of 6.

**FW\_CRYPTO\_KEY\_EXCHANGE\_MAX\_V2\_10 = FW\_CRYPTO\_KEY\_EXCHANGE\_DH24:** This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x020A and earlier. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 6.

**FW\_CRYPTO\_KEY\_EXCHANGE\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 7.

### FW\_CRYPTO\_ENCRYPTION\_TYPE

This enumeration is used to identify supported encryption algorithms.

1. typedef enum \_tag\_FW\_CRYPTO\_ENCRYPTION\_TYPE
2. {
3. FW\_CRYPTO\_ENCRYPTION\_NONE = 0,
4. FW\_CRYPTO\_ENCRYPTION\_DES = 1,
5. FW\_CRYPTO\_ENCRYPTION\_3DES = 2,
6. FW\_CRYPTO\_ENCRYPTION\_AES128 = 3,
7. FW\_CRYPTO\_ENCRYPTION\_AES192 = 4,
8. FW\_CRYPTO\_ENCRYPTION\_AES256 = 5,
9. FW\_CRYPTO\_ENCRYPTION\_AES\_GCM128 = 6,
10. FW\_CRYPTO\_ENCRYPTION\_AES\_GCM192 = 7,
11. FW\_CRYPTO\_ENCRYPTION\_AES\_GCM256 = 8,
12. FW\_CRYPTO\_ENCRYPTION\_MAX = 9,
13. FW\_CRYPTO\_ENCRYPTION\_MAX\_V2\_0 = FW\_CRYPTO\_ENCRYPTION\_AES\_GCM128
14. } FW\_CRYPTO\_ENCRYPTION\_TYPE;

**FW\_CRYPTO\_ENCRYPTION\_NONE:** This value MUST be used only when no encryption is to be performed. This is a valid value. This symbolic constant has a value of 0.

**FW\_CRYPTO\_ENCRYPTION\_DES:** Uses the DES algorithm for encryption. This symbolic constant has a value of 1.

**FW\_CRYPTO\_ENCRYPTION\_3DES:** Uses the 3DES algorithm for encryption. This symbolic constant has a value of 2.

**FW\_CRYPTO\_ENCRYPTION\_AES128:** Uses the AES algorithm with a 128-bit key size for encryption. This symbolic constant has a value of 3.

**FW\_CRYPTO\_ENCRYPTION\_AES192:** Uses the AES algorithm with a 192-bit key size for encryption. This symbolic constant has a value of 4.

**FW\_CRYPTO\_ENCRYPTION\_AES256:** Uses the AES algorithm with a 256-bit key size for encryption. This symbolic constant has a value of 5.

**FW\_CRYPTO\_ENCRYPTION\_AES\_GCM128:** Uses the AESGCM algorithm with a 128-bit key size for encryption. This symbolic constant has a value of 6.

**FW\_CRYPTO\_ENCRYPTION\_AES\_GCM192:** Uses the AESGCM algorithm with a 192-bit key size for encryption. This symbolic constant has a value of 7.

**FW\_CRYPTO\_ENCRYPTION\_AES\_GCM256:** Uses the AESGCM algorithm with a 256-bit key size for encryption. This symbolic constant has a value of 8.

**FW\_CRYPTO\_ENCRYPTION\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 9.

**FW\_CRYPTO\_ENCRYPTION\_MAX\_V2\_0:** For schema version 0x0200, this value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x0200 and earlier. It is defined for simplicity in writing IDL definitions and describing semantic checks against policy schema versions of 0x0200. This symbolic constant has a value of 6.

### FW\_CRYPTO\_HASH\_TYPE

This enumeration is used to identify the different hashing (integrity protection) algorithms supported.

1. typedef enum \_tag\_FW\_CRYPTO\_HASH\_TYPE
2. {
3. FW\_CRYPTO\_HASH\_NONE = 0,
4. FW\_CRYPTO\_HASH\_MD5 = 1,
5. FW\_CRYPTO\_HASH\_SHA1 = 2,
6. FW\_CRYPTO\_HASH\_SHA256 = 3,
7. FW\_CRYPTO\_HASH\_SHA384 = 4,
8. FW\_CRYPTO\_HASH\_AES\_GMAC128 = 5,
9. FW\_CRYPTO\_HASH\_AES\_GMAC192 = 6,
10. FW\_CRYPTO\_HASH\_AES\_GMAC256 = 7,
11. FW\_CRYPTO\_HASH\_MAX = 8,
12. FW\_CRYPTO\_HASH\_MAX\_V2\_0 = FW\_CRYPTO\_HASH\_SHA256
13. } FW\_CRYPTO\_HASH\_TYPE;

**FW\_CRYPTO\_HASH\_NONE:** This value MUST be used only when no hashing is to be performed. This is a valid value. This symbolic constant has a value of 0.

**FW\_CRYPTO\_HASH\_MD5:** Use the MD5 algorithm for hashing (integrity protection). This symbolic constant has a value of 1.

**FW\_CRYPTO\_HASH\_SHA1:** Use the SHA1 algorithm for hashing (integrity protection). This symbolic constant has a value of 2.

**FW\_CRYPTO\_HASH\_SHA256:** Use the SHA256 algorithm for hashing (integrity protection). This symbolic constant has a value of 3.

**FW\_CRYPTO\_HASH\_SHA384:** Use the SHA384 algorithm for hashing (integrity protection). This symbolic constant has a value of 4.

**FW\_CRYPTO\_HASH\_AES\_GMAC128:** Use the AESGMAC128 algorithm for hashing (integrity protection). This symbolic constant has a value of 5.

**FW\_CRYPTO\_HASH\_AES\_GMAC192:** Use the AESGMAC192 algorithm for hashing (integrity protection). This symbolic constant has a value of 6.

**FW\_CRYPTO\_HASH\_AES\_GMAC256:** Use the AESGMAC256 algorithm for hashing (integrity protection). This symbolic constant has a value of 7.

**FW\_CRYPTO\_HASH\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 8.

**FW\_CRYPTO\_HASH\_MAX\_V2\_0:** This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x0200 and earlier. It is defined for simplicity in writing IDL definitions and describing semantic checks against policy schema versions of 0x0200. This symbolic constant has a value of 3.

### FW\_CRYPTO\_PROTOCOL\_TYPE

This enumeration is used to identify the different combinations of supported [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) enforcement protocols.

1. typedef enum \_tag\_FW\_CRYPTO\_PROTOCOL\_TYPE
2. {
3. FW\_CRYPTO\_PROTOCOL\_INVALID = 0,
4. FW\_CRYPTO\_PROTOCOL\_AH = 1,
5. FW\_CRYPTO\_PROTOCOL\_ESP = 2,
6. FW\_CRYPTO\_PROTOCOL\_BOTH = 3,
7. FW\_CRYPTO\_PROTOCOL\_AUTH\_NO\_ENCAP = 4,
8. FW\_CRYPTO\_PROTOCOL\_MAX = 5,
9. FW\_CRYPTO\_PROTOCOL\_MAX\_2\_1 = (FW\_CRYPTO\_PROTOCOL\_BOTH + 1)
10. } FW\_CRYPTO\_PROTOCOL\_TYPE;

**FW\_CRYPTO\_PROTOCOL\_INVALID:** This value MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 0.

**FW\_CRYPTO\_PROTOCOL\_AH:** Uses the [**authentication header (AH)**](#gt_efa9e2b6-53fb-473e-8824-e276ebde4c97) to enforce IPsec. This symbolic constant has a value of 1.

**FW\_CRYPTO\_PROTOCOL\_ESP:** Uses the [**ESP**](#gt_430b4a39-0b2c-402f-847d-e6a8520934c7) protocol header. This symbolic constant has a value of 2.

**FW\_CRYPTO\_PROTOCOL\_BOTH:** Uses both the AH and ESP protocol headers. This symbolic constant has a value of 3.

**FW\_CRYPTO\_PROTOCOL\_AUTH\_NO\_ENCAP:** Uses no encapsulation. This sends the first packet twice: once by using an ESP header and again without any header; subsequent packets have no additional headers. This symbolic constant has a value of 4.

**FW\_CRYPTO\_PROTOCOL\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 5.

**FW\_CRYPTO\_PROTOCOL\_MAX\_2\_1:** This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x0201 and earlier. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 4.

### FW\_PHASE1\_CRYPTO\_SUITE

This structure describes an [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) Phase 1 (or main mode) cryptographic suite. A cryptographic suite is a proposal of a set of algorithms and parameters that specify how different types of enforcement and protection are suggested to be performed.

1. typedef struct \_tag\_FW\_PHASE1\_CRYPTO\_SUITE {
2. [range(FW\_CRYPTO\_KEY\_EXCHANGE\_NONE, FW\_CRYPTO\_KEY\_EXCHANGE\_MAX-1)]
3. FW\_CRYPTO\_KEY\_EXCHANGE\_TYPE KeyExchange;
4. [range(FW\_CRYPTO\_ENCRYPTION\_NONE+1, FW\_CRYPTO\_ENCRYPTION\_MAX-1)]
5. FW\_CRYPTO\_ENCRYPTION\_TYPE Encryption;
6. [range(FW\_CRYPTO\_HASH\_NONE+1, FW\_CRYPTO\_HASH\_MAX-1)]
7. FW\_CRYPTO\_HASH\_TYPE Hash;
8. unsigned long dwP1CryptoSuiteFlags;
9. } FW\_PHASE1\_CRYPTO\_SUITE,
10. \*PFW\_PHASE1\_CRYPTO\_SUITE;

**KeyExchange:**  This field is of type [FW\_CRYPTO\_KEY\_EXCHANGE\_TYPE](#Section_b400008796b046c7b15209a55d985f0f). It specifies the key exchange algorithm for this suite proposal.

**Encryption:**  This field is of type [FW\_CRYPTO\_ENCRYPTION\_TYPE](#Section_a74eafcd19ff4305b1050952ae20a2dd). It specifies the encryption algorithm for this suite proposal.

**Hash:**  This field is of type [FW\_CRYPTO\_HASH\_TYPE](#Section_eae09dba8bed43e385bddce8497de9a1). It specifies the hash (integrity protection) algorithm for this suite proposal.

**dwP1CryptoSuiteFlags:**  This is a reserved value and is not used. It MUST be set to 0x00000000.

### FW\_PHASE2\_CRYPTO\_SUITE

This structure describes an [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) Phase 2 (or quick mode) cryptographic suite. A cryptographic suite is a proposal of a set of algorithms and parameters that specify how different types of enforcement and protection are suggested to be performed. It also suggests timeouts for which a key is valid and at which re-keying operations should be performed.

1. typedef struct \_tag\_FW\_PHASE2\_CRYPTO\_SUITE {
2. [range(FW\_CRYPTO\_PROTOCOL\_INVALID+1,FW\_CRYPTO\_PROTOCOL\_MAX-1)]
3. FW\_CRYPTO\_PROTOCOL\_TYPE Protocol;
4. FW\_CRYPTO\_HASH\_TYPE AhHash;
5. FW\_CRYPTO\_HASH\_TYPE EspHash;
6. FW\_CRYPTO\_ENCRYPTION\_TYPE Encryption;
7. unsigned long dwTimeoutMinutes;
8. unsigned long dwTimeoutKBytes;
9. unsigned long dwP2CryptoSuiteFlags;
10. } FW\_PHASE2\_CRYPTO\_SUITE,
11. \*PFW\_PHASE2\_CRYPTO\_SUITE;

**Protocol:**  This field is of type [FW\_CRYPTO\_PROTOCOL\_TYPE](#Section_d97bac3603e842159984f2fbddd66be0), and it specifies the IPsec enforcement protocol combination suggested for this suite.

**AhHash:**  This field is of type [FW\_CRYPTO\_HASH\_TYPE](#Section_eae09dba8bed43e385bddce8497de9a1). It specifies the hash (integrity protection) algorithm for this suite proposal when using the [**authentication header**](#gt_efa9e2b6-53fb-473e-8824-e276ebde4c97) protocol.

**EspHash:**  This field is of type FW\_CRYPTO\_HASH\_TYPE. It specifies the hash (integrity protection) algorithm for this suite proposal when using the [**ESP**](#gt_430b4a39-0b2c-402f-847d-e6a8520934c7) protocol.

**Encryption:**  This field is of type [FW\_CRYPTO\_ENCRYPTION\_TYPE](#Section_a74eafcd19ff4305b1050952ae20a2dd). It specifies the encryption algorithm for this suite proposal.

**dwTimeoutMinutes:**  This is the timeout or lifetime of the key used in this proposal defined in minutes.

**dwTimeoutKBytes:**  This is the timeout or lifetime of the key used in this proposal defined in kilobytes processed with this configuration.

**dwP2CryptoSuiteFlags:**  This field is reserved and is not used. It MUST be set to 0x00000000.

The following are semantic validation checks that Phase 2 cryptographic suites MUST pass:

* The **dwTimeoutMinutes** field MUST be greater than or equal to 5 and less than or equal to 2,879.
* The **dwTimeoutKBytes** field MUST be greater than or equal to 20,480 and less than or equal to 2,147,483,647.
* If the **Protocol** field is FW\_CRYPTO\_PROTOCOL\_AH or FW\_CRYPTO\_PROTOCOL\_BOTH, the **AhHash** field MUST NOT be equal to FW\_CRYPTO\_HASH\_NONE.
* If the **Protocol** field is FW\_CRYPTO\_PROTOCOL\_BOTH, the **AhHash** field MUST be equal to the **EspHash** field.
* If the **Protocol** field is FW\_CRYPTO\_PROTOCOL\_BOTH or FW\_CRYPTO\_PROTOCOL\_ESP, **EspHash** MUST NOT be set to FW\_CRYPTO\_HASH\_NONE or **Encryption** MUST NOT be set to FW\_CRYPTO\_ENCRYPTION\_NONE, but not both.

### FW\_PHASE1\_CRYPTO\_FLAGS

This enumeration is used to identify the different cryptographic flags that are supported.

1. typedef enum \_tag\_FW\_PHASE1\_CRYPTO\_FLAGS
2. {
3. FW\_PHASE1\_CRYPTO\_FLAGS\_NONE = 0x00,
4. FW\_PHASE1\_CRYPTO\_FLAGS\_DO\_NOT\_SKIP\_DH = 0x01,
5. FW\_PHASE1\_CRYPTO\_FLAGS\_MAX = 0x02
6. } FW\_PHASE1\_CRYPTO\_FLAGS;

**FW\_PHASE1\_CRYPTO\_FLAGS\_NONE:** This value represents no flag. It is used when none of the behaviors that are represented by the defined flags in the enumeration are intended. This symbolic constant has a value of 0x00.

**FW\_PHASE1\_CRYPTO\_FLAGS\_DO\_NOT\_SKIP\_DH:** This flag ensures that [**Authenticated IP (AuthIP)**](#gt_3791f3e1-cf2f-4605-9fcc-54f526f036cf), as specified in [[MS-AIPS]](%5bMS-AIPS%5d.pdf#Section_eee3de6438474451978e9513ff187d30), always performs a DH key exchange. (AuthIP can avoid this exchange because the protocol already contains enough key material information to protect the negotiation. Hence, by skipping DH, round trips and the computational cost of DH are avoided.) This symbolic constant has a value of 0x01.

**FW\_PHASE1\_CRYPTO\_FLAGS\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 0x02.

### FW\_PHASE2\_CRYPTO\_PFS

This enumeration is used to identify the different [**perfect forward secrecy (PFS)**](#gt_5d8948bc-5e32-483b-906d-42f785d0df18) options supported.

1. typedef enum \_tag\_FW\_PHASE2\_CRYPTO\_PFS
2. {
3. FW\_PHASE2\_CRYPTO\_PFS\_INVALID = 0,
4. FW\_PHASE2\_CRYPTO\_PFS\_DISABLE = 1,
5. FW\_PHASE2\_CRYPTO\_PFS\_PHASE1 = 2,
6. FW\_PHASE2\_CRYPTO\_PFS\_DH1 = 3,
7. FW\_PHASE2\_CRYPTO\_PFS\_DH2 = 4,
8. FW\_PHASE2\_CRYPTO\_PFS\_DH2048 = 5,
9. FW\_PHASE2\_CRYPTO\_PFS\_ECDH256 = 6,
10. FW\_PHASE2\_CRYPTO\_PFS\_ECDH384 = 7,
11. FW\_PHASE2\_CRYPTO\_PFS\_DH24 = 8,
12. FW\_PHASE2\_CRYPTO\_PFS\_MAX\_V2\_10 = FW\_PHASE2\_CRYPTO\_PFS\_DH24 = FW\_PHASE2\_CRYPTO\_PFS\_DH24,
13. FW\_PHASE2\_CRYPTO\_PFS\_MAX = 9
14. } FW\_PHASE2\_CRYPTO\_PFS;

**FW\_PHASE2\_CRYPTO\_PFS\_INVALID:** This value MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 0.

**FW\_PHASE2\_CRYPTO\_PFS\_DISABLE:** Do not renegotiate; instead, reuse the keying material negotiated in Phase 1 (main mode). This symbolic constant has a value of 1.

**FW\_PHASE2\_CRYPTO\_PFS\_PHASE1:** Use Phase 1 key exchange to negotiate a Phase 2 (quick mode) key for every Phase 2 negotiation. This symbolic constant has a value of 2.

**FW\_PHASE2\_CRYPTO\_PFS\_DH1:** Use DH1 key exchange to negotiate a Phase 2 (quick mode) key for every Phase 2 negotiation. This symbolic constant has a value of 3.

**FW\_PHASE2\_CRYPTO\_PFS\_DH2:** Use DH2 key exchange to negotiate a Phase 2 (quick mode) key for every Phase 2 negotiation. This symbolic constant has a value of 4.

**FW\_PHASE2\_CRYPTO\_PFS\_DH2048:** Use DH2048 key exchange to negotiate a Phase 2 (quick mode) key for every Phase 2 negotiation. This symbolic constant has a value of 5.

**FW\_PHASE2\_CRYPTO\_PFS\_ECDH256:** Use ECDH256 key exchange to negotiate a Phase 2 (quick mode) key for every Phase 2 negotiation. This symbolic constant has a value of 6.

**FW\_PHASE2\_CRYPTO\_PFS\_ECDH384:** Use ECDH384 key exchange to negotiate a Phase 2 (quick mode) key for every Phase 2 negotiation. This symbolic constant has a value of 7.

**FW\_PHASE2\_CRYPTO\_PFS\_DH24:** Use DH24 key exchange to negotiate a Phase 2 (quick mode) key for every Phase 2 negotiation. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used. This symbolic constant has a value of 8.

**FW\_PHASE2\_CRYPTO\_PFS\_MAX\_V2\_10 = FW\_PHASE2\_CRYPTO\_PFS\_DH24:** This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x020A and earlier. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 8.

**FW\_PHASE2\_CRYPTO\_PFS\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 9.

### FW\_CRYPTO\_SET

This structure contains a list of cryptographic suite elements that are ordered from highest to lowest preference and are negotiated with remote peers to establish cryptographic protection algorithms.

1. typedef struct \_tag\_FW\_CRYPTO\_SET {
2. struct \_tag\_FW\_CRYPTO\_SET\* pNext;
3. unsigned short wSchemaVersion;
4. [range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)]
5. FW\_IPSEC\_PHASE IpSecPhase;
6. [string, range(1,255), ref] wchar\_t\* wszSetId;
7. [string, range(1,10001)] wchar\_t\* wszName;
8. [string, range(1,10001)] wchar\_t\* wszDescription;
9. [string, range(1,10001)] wchar\_t\* wszEmbeddedContext;
10. [switch\_type(FW\_IPSEC\_PHASE), switch\_is(IpSecPhase)]
11. union {
12. [case(FW\_IPSEC\_PHASE\_1)]
13. struct {
14. unsigned short wFlags;
15. [range(0,1000)] unsigned long dwNumPhase1Suites;
16. [size\_is(dwNumPhase1Suites)] PFW\_PHASE1\_CRYPTO\_SUITE pPhase1Suites;
17. unsigned long dwTimeoutMinutes;
18. unsigned long dwTimeoutSessions;
19. };
20. [case(FW\_IPSEC\_PHASE\_2)]
21. struct {
22. FW\_PHASE2\_CRYPTO\_PFS Pfs;
23. [range(0,1000)] unsigned long dwNumPhase2Suites;
24. [size\_is(dwNumPhase2Suites)] PFW\_PHASE2\_CRYPTO\_SUITE pPhase2Suites;
25. };
26. };
27. [range(FW\_RULE\_ORIGIN\_INVALID,FW\_RULE\_ORIGIN\_MAX-1)]
28. FW\_RULE\_ORIGIN\_TYPE Origin;
29. [string, range(1,10001)] wchar\_t\* wszGPOName;
30. FW\_RULE\_STATUS Status;
31. unsigned long dwCryptoSetFlags;
32. } FW\_CRYPTO\_SET,
33. \*PFW\_CRYPTO\_SET;

**pNext:**  A pointer to the next FW\_CRYPTO\_SET in the list.

**wSchemaVersion:**  Specifies the version of the set.

**IpSecPhase:**  This field is of type [FW\_IPSEC\_PHASE](#Section_e23d75abc5f14b77b290f0e0e25d72e1), and it specifies if this cryptographic set applies for Phase1 (main mode) or Phase2 (quick mode).

**wszSetId:**  A pointer to a [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string that uniquely identifies the set. The primary set for this policy object is identified with the "{E5A5D32A-4BCE-4e4d-B07F-4AB1BA7E5FE1}" string for Phase1 and with the "{E5A5D32A-4BCE-4e4d-B07F-4AB1BA7E5FE2}" string for Phase2.

**wszName:**  A pointer to a Unicode string that provides a friendly name for the set.

**wszDescription:**  A pointer to a Unicode string that provides a friendly description for the set.

**wszEmbeddedContext:**  A pointer to a Unicode string. A client implementation MAY use this field to store implementation-specific client context. The server MUST NOT interpret the value of this string. The server MUST preserve the value of this string unmodified.

**wFlags:**  This field is a combination of the [FW\_PHASE1\_CRYPTO\_FLAGS](#Section_6f3d76aab2424f2b8c5ff9ccbbc04c53) enumeration bit flags.

**dwNumPhase1Suites:**  Specifies the number of Phase1 suites that the structure contains.

**pPhase1Suites:**  A pointer to an array of **dwNumPhase1Suites** contiguous [FW\_PHASE1\_CRYPTO\_SUITE](#Section_cd273bb434214831a979347630eb3b16) elements.

**dwTimeoutMinutes:**  This value is a lifetime in minutes before a Phase1 established key is renegotiated.

**dwTimeoutSessions:**  This value is the number of sessions before a Phase1 established key is renegotiated.

**Pfs:**  This field MUST contain a valid value of those in the [FW\_PHASE2\_CRYPTO\_PFS](#Section_a281cd9c0d3e40718a4fd0b5ada838a2) enumeration. It describes the [**perfect forward secrecy**](#gt_5d8948bc-5e32-483b-906d-42f785d0df18) used for quick mode cryptographic operations.

**dwNumPhase2Suites:**  Specifies the number of Phase2 suites that the structure contains.

**pPhase2Suites:**  A pointer to an array of [FW\_PHASE2\_CRYPTO\_SUITE](#Section_99a997b6de524648977ce9efd6058254) elements. The number of elements is given by **dwNumPhase2Suites**.

**Origin:**  This field is the set origin, as specified in the [FW\_RULE\_ORIGIN\_TYPE](#Section_9d295321d75c41c0ab0d7a78df40f77c) enumeration. It MUST be filled on enumerated rules and ignored on input.

**wszGPOName:**  A pointer to a Unicode string containing the displayName of the [**GPO**](#gt_dec32233-8776-4151-91a0-8624a2b9abb0) containing this object. When adding a new object, this field is not used. The client SHOULD set the value to NULL, and the server MUST ignore the value. When enumerating an existing object, if the client does not set the FW\_ENUM\_RULES\_FLAG\_RESOLVE\_GPO\_NAME flag, the server MUST set the value to NULL. Otherwise, the server MUST set the value to the displayName of the GPO containing the object or NULL if the object is not contained within a GPO. For details about how the server initializes an object from a GPO, see section [3.1.3](#Section_e8924ac5aa4a41d1bf654f46b3d399aa). For details about how the displayName of a GPO is stored, see [[MS-GPOL]](%5bMS-GPOL%5d.pdf#Section_62d1292462524052996f161d2b9019f4) section 2.3.

**Status:**  The status code of the set, as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out when the structure is returned as output. On input, this field MUST be set to FW\_RULE\_STATUS\_OK.

**dwCryptoSetFlags:**  Bit flags from FW\_CRYPTO\_SET\_FLAGS.

The following are semantic checks that cryptographic sets MUST pass:

* The **wSchemaVersion** field MUST NOT be less than 0x000200.
* The **wszSetId** field MUST NOT contain the pipe (|) character, MUST NOT be NULL, MUST be a string at least 1 character long, and MUST NOT be greater than or equal to 255 characters.
* If the **wszName** field string is not NULL, it MUST be at least 1 character long, MUST NOT be greater than or equal to 10,000 characters, and MUST NOT contain the pipe (|) character.
* If the **wszDescription** field string is not NULL, it MUST be at least 1 character long, MUST NOT be greater than or equal to 10,000 characters, and MUST NOT contain the pipe (|) character.
* If the **wszEmbeddedContext** field string is not NULL, it MUST be at least 1 character long, MUST NOT be greater than or equal to 10,000 characters, and MUST NOT contain the pipe (|) character.
* The **IpSecPhase** field MUST have valid FW\_IPSEC\_PHASE values.
* If the **IpSecPhase** field is FW\_IPSEC\_PHASE\_1:
  + The **wszSetId** field MUST be equal to the primary Phase1 cryptographic set ID. (There is only one Phase1 cryptographic set allowed per store.)
  + The **wFlags** field of the set MUST NOT be greater than or equal to FW\_PHASE1\_CRYPTO\_FLAGS\_MAX.
  + The **dwTimeoutMinutes** field of the set MUST be greater than or equal to 1, and MUST be less than or equal to 2,879.
  + The **dwTimeoutSessions** field of the set MUST be less than or equal to 2,147,483,647.
  + The cryptographic set MUST have at least one Phase1 cryptographic suite.
  + The **pPhase1Suites** array MUST contain exactly **dwNumPhase1Suites** entries.
  + All cryptographic suites within the set MUST have the same value in the **KeyExchange** field and MUST have valid values.
  + All Phase1 suites MUST NOT have a **KeyExchange** field with the FW\_CRYPTO\_ENCRYPTION\_INVALID value and MUST have valid values.
  + If the set has a schema policy version of 0x0200, all Phase1 suites MUST NOT have an **Encryption** field with values greater than or equal to FW\_CRYPTO\_ENCRYPTION\_MAX\_V2\_0.
  + All Phase1 suites MUST NOT have an **Encryption** field with the FW\_CRYPTO\_ENCRYPTION\_NONE value and MUST have valid values less than FW\_CRYPTO\_ENCRYPTION\_MAX\_V2\_0.
  + If the set has a schema policy version of 0x0200, all Phase1 suites MUST NOT have a **Hash** field that has values greater than or equal to FW\_CRYPTO\_HASH\_MAX\_V2\_0.
  + All Phase1 suites MUST NOT have a **Hash** field that has the FW\_CRYPTO\_HASH\_NONE value and MUST have either MD5 (FW\_CRYPTO\_HASH\_MD5) or SHA (FW\_CRYPTO\_HASH\_SHA1, FW\_CRYPTO \_HASH\_SHA256, FW\_CRYPTO\_HASH\_SHA384) valid values.
* If the **IpSecPhase** field is FW\_IPSEC\_PHASE\_2:
  + The **wszSetId** field MUST NOT have the primary Phase2 cryptographic set ID as a prefix.
  + The cryptographic set MUST have at least one Phase2 cryptographic suite.
  + The **pPhase2Suites** array MUST contain exactly **dwNumPhase2Suites** entries.
  + The **Pfs** field MUST NOT be FW\_PHASE2\_CRYPTO\_PFS\_INVALID and MUST have valid values.
  + If the set has a schema policy version of 0x0200, all Phase2 cryptographic suites MUST NOT have an **AhHash** field or **EspHash** field with values greater than or equal to FW\_CRYPTO\_HASH\_MAX\_V2\_0.
  + If the set has a schema policy version of 0x0200, all Phase2 suites MUST NOT have an **Encryption** field with values greater than or equal to FW\_CRYPTO\_ENCRYPTION\_MAX\_V2\_0.
  + All Phase2 suites within the set MUST NOT have a **dwTimeoutMinutes** field less than FW\_MIN\_CRYPTO\_PHASE2\_TIMEOUT\_MINUTES (5) or greater than FW\_MAX\_CRYPTO\_PHASE2\_TIMEOUT\_MINUTES (48 \* 60 -1).
  + All Phase2 suites within the set MUST NOT have a **dwTimeoutKBytes** field of less than FW\_MIN\_CRYPTO\_PHASE2\_TIMEOUT\_KBYTES (20480) or greater than FW\_MAX\_CRYPTO\_PHASE2\_TIMEOUT\_KBYTES (2147483647).
  + All the Phase2 suites within the set MUST NOT have a **Protocol** field with FW\_CRYPTO\_PROTOCOL\_INVALID and MUST have valid values.
  + For all suites that have the **Protocol** field equal to FW\_CRYPTO\_PROTOCOL\_AH or to FW\_CRYPTO\_PROTOCOL\_BOTH:
    - All suites MUST NOT have an **AhHash** field with the FW\_CRYPTO\_HASH\_NONE value, and MUST have valid values not equal to FW\_CRYPTO\_HASH\_SHA384.
  + For all suites that have the **Protocol** field equal to FW\_CRYPTO\_PROTOCOL\_BOTH:
    - All suites MUST have the **AhHash** field equal to the **EspHash** field.
  + For all suites that have the **Protocol** field equal to FW\_CRYPTO\_PROTOCOL\_ESP:
    - All suites MUST have an **EspHash** field with valid values, including FW\_CRYPTO\_HASH\_NONE. The **EspHash** field MUST NOT equal FW\_CRYPTO\_HASH\_SHA384.
    - All suites MUST have an **Encryption** field with valid values, including FW\_CRYPTO\_ENCRYPTION\_NONE.
    - All suites MUST not have both the **EspHash** field equal to FW\_CRYPTO\_HASH\_NONE and the **Encryption** field equal to FW\_CRYPTO\_ENCRYPTION\_NONE.
    - All suites that have the **Encryption** field equal to FW\_CRYPTO\_ENCRYPTION\_AES\_GCM128, 192, or 256 MUST also have a corresponding FW\_CRYPTO\_HASH\_AES\_GMAC128, 192, or 256 value on the **EspHash** field. An AES GCM encryption algorithm corresponds to an AES GMAC hash algorithm if both use the same bit size.

### FW\_BYTE\_BLOB

This structure contains a memory section. The format of the memory is defined by the context where it is used; for example, see the **SubjectName** field of the [FW\_CERT\_INFO](#Section_dfc16343eb514bbab890df5a58839b2b) structure.

1. typedef struct \_tag\_FW\_BYTE\_BLOB {
2. [range(0,10000)] unsigned long dwSize;
3. [size\_is(dwSize)] unsigned char\* Blob;
4. } FW\_BYTE\_BLOB,
5. \*PFW\_BYTE\_BLOB;

**dwSize:**  This field specifies the size in octets of the **Blob** field.

**Blob:**  A pointer to an array of **dwSize** octets.

### FW\_COOKIE\_PAIR

This structure holds random numbers generated out of [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) negotiations.

1. typedef struct \_tag\_FW\_COOKIE\_PAIR {
2. unsigned \_\_int64 Initiator;
3. unsigned \_\_int64 Responder;
4. } FW\_COOKIE\_PAIR,
5. \*PFW\_COOKIE\_PAIR;

**Initiator:**  A random number that maps to the negotiated state that is a [**security association**](#gt_67cbf867-7a49-41f3-a68f-37b5f9035acb) of the machine that initiated communication and, hence, initiated [**IKE**](#gt_294fef97-5790-4d41-971e-dd255b783e68)/[**AuthIP**](#gt_3791f3e1-cf2f-4605-9fcc-54f526f036cf) (for more information, see [[RFC2409]](https://go.microsoft.com/fwlink/?LinkId=90349)) as specified in [[MS-IKEE]](%5bMS-IKEE%5d.pdf#Section_e05e2762179a4c3fbfb50aca7bbefe79) and [[MS-AIPS]](%5bMS-AIPS%5d.pdf#Section_eee3de6438474451978e9513ff187d30) traffic.

**Responder:**  A random number that maps to the negotiated state that is a security association of the machine that responded to the communication and, hence, responded to the IKE/AuthIP traffic.

### FW\_PHASE1\_KEY\_MODULE\_TYPE

This enumeration identifies the different [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) Key Exchange negotiation protocols that can be used.

1. typedef enum \_tag\_FW\_PHASE1\_KEY\_MODULE\_TYPE
2. {
3. FW\_PHASE1\_KEY\_MODULE\_INVALID = 0,
4. FW\_PHASE1\_KEY\_MODULE\_IKE = 1,
5. FW\_PHASE1\_KEY\_MODULE\_AUTH\_IP = 2,
6. FW\_PHASE1\_KEY\_MODULE\_MAX = 3
7. } FW\_PHASE1\_KEY\_MODULE\_TYPE;

**FW\_PHASE1\_KEY\_MODULE\_INVALID:** The FW\_PHASE1\_KEY\_MODULE\_INVALID constant MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 0.

**FW\_PHASE1\_KEY\_MODULE\_IKE:** The keying protocol was [**IKE**](#gt_294fef97-5790-4d41-971e-dd255b783e68). This symbolic constant has a value of 1.

**FW\_PHASE1\_KEY\_MODULE\_AUTH\_IP:** The keying protocol was [**AuthIP**](#gt_3791f3e1-cf2f-4605-9fcc-54f526f036cf). This symbolic constant has a value of 2.

**FW\_PHASE1\_KEY\_MODULE\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 3.

### FW\_CERT\_INFO

This structure represents information on the certificate used in the certificate-based authentication mechanisms.

1. typedef struct \_tag\_FW\_CERT\_INFO {
2. FW\_BYTE\_BLOB SubjectName;
3. [range(FW\_AUTH\_SUITE\_FLAGS\_NONE, FW\_AUTH\_SUITE\_FLAGS\_MAX-1)]
4. unsigned long dwCertFlags;
5. } FW\_CERT\_INFO,
6. \*PFW\_CERT\_INFO;

**SubjectName:**  The subject name of the certificate represented as a [FW\_BYTE\_BLOB](#Section_8e7e10f0590043618403d8833934c491) type. This BLOB is an ASN.1-encoded sequence of RDN attributes.

**dwCertFlags:**  This field can be a combination of bit flags from [FW\_AUTH\_SUITE\_FLAGS](#Section_ff03b48b87cf4020bf0b7f4c680d9b02). This field MUST use only health certificate or certificate to account mapping flags, which represent certificate characteristics.

### FW\_AUTH\_INFO

This structure contains information on the local and remote hosts that resulted from the authentication methods performed between them.

1. typedef struct \_tag\_FW\_AUTH\_INFO {
2. [range(FW\_AUTH\_METHOD\_INVALID + 1, FW\_AUTH\_METHOD\_MAX)]
3. FW\_AUTH\_METHOD AuthMethod;
4. [switch\_type(FW\_AUTH\_METHOD), switch\_is(AuthMethod)]
5. union {
6. [case(FW\_AUTH\_METHOD\_MACHINE\_CERT,FW\_AUTH\_METHOD\_USER\_CERT)]
7. struct {
8. FW\_CERT\_INFO MyCert;
9. FW\_CERT\_INFO PeerCert;
10. };
11. [case(FW\_AUTH\_METHOD\_MACHINE\_KERB,FW\_AUTH\_METHOD\_USER\_KERB, FW\_AUTH\_METHOD\_MACHINE\_NEGOEX,FW\_AUTH\_METHOD\_USER\_NEGOEX)]
12. struct {
13. [string, range(1,10001)] wchar\_t\* wszMyId;
14. [string, range(1,10001)] wchar\_t\* wszPeerId;
15. };
16. [default]  ;
17. };
18. unsigned long dwAuthInfoFlags;
19. } FW\_AUTH\_INFO,
20. \*PFW\_AUTH\_INFO;

**AuthMethod:**  This field contains the authentication method used to establish the identities of the [**endpoints**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee) and is stored in the [**security association**](#gt_67cbf867-7a49-41f3-a68f-37b5f9035acb). The field can take valid values from the [FW\_AUTH\_METHOD](#Section_59e71d46440747d29cf58889fd3a74f2) enumeration.

**MyCert:**  This field contains the subject name and certification flags (health, account mapping, exclude [**CA**](#gt_c925d5d7-a442-4ba4-9586-5f94ccec847a)) from the certificate of the local host that was used in the authentication process when a certificate-based authentication method is used.

**PeerCert:**  This field contains the subject name and certification flags (health, account mapping, exclude CA) from the certificate of the remote host that was used in the authentication process when a certificate-based authentication method is used.

**wszMyId:**  A pointer to a [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string representing the identity of the local host when a Kerberos-based authentication method, as specified in [[MS-KILE]](%5bMS-KILE%5d.pdf#Section_2a32282edd484ad9a542609804b02cc9), is used.

**wszPeerId:**  A pointer to a Unicode string representing the identity of the remote host when a Kerberos-based authentication method, as specified in [MS-KILE], is used.

**dwAuthInfoFlags:**  Reserved value and not currently used. It MUST be set to 0.

### FW\_ENDPOINTS

This structure represents the two [**endpoints**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee), source and destination, that participate in IP communication.

1. typedef struct \_tag\_FW\_ENDPOINTS {
2. [range(FW\_IP\_VERSION\_INVALID+1,FW\_IP\_VERSION\_MAX-1)]
3. FW\_IP\_VERSION IpVersion;
4. unsigned long dwSourceV4Address;
5. unsigned long dwDestinationV4Address;
6. unsigned char SourceV6Address[16];
7. unsigned char DestinationV6Address[16];
8. } FW\_ENDPOINTS,
9. \*PFW\_ENDPOINTS;

**IpVersion:**  This field specifies the Internet Protocol version used. This field MUST contain a valid value from the [FW\_IP\_VERSION](#Section_f0350f85904844268264fe11d62c5af3) enumeration.

**dwSourceV4Address:**  This field is the IPv4 address of the source endpoint.

**dwDestinationV4Address:**  This field is the IPv4 address of the destination endpoint.

**SourceV6Address:**  This field is a 16-octet array that represents the IPv6 address of the source endpoint.

**DestinationV6Address:**  This field is a 16-octet array that represents the IPv6 address of the destination endpoint.

The v4 versions or the v6 versions of the fields are used depending on the **IpVersion** field value.

### FW\_PHASE1\_SA\_DETAILS

This structure represents a [**security association**](#gt_67cbf867-7a49-41f3-a68f-37b5f9035acb) that is established after the main mode negotiations take place; it contains the selected algorithms to enforce [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) and the methods and results of the authentication process.

1. typedef struct \_tag\_FW\_PHASE1\_SA\_DETAILS {
2. unsigned \_\_int64 SaId;
3. [range( FW\_PHASE1\_KEY\_MODULE\_INVALID+1,FW\_PHASE1\_KEY\_MODULE\_MAX-1)]
4. FW\_PHASE1\_KEY\_MODULE\_TYPE KeyModuleType;
5. FW\_ENDPOINTS Endpoints;
6. FW\_PHASE1\_CRYPTO\_SUITE SelectedProposal;
7. unsigned long dwProposalLifetimeKBytes;
8. unsigned long dwProposalLifetimeMinutes;
9. unsigned long dwProposalMaxNumPhase2;
10. FW\_COOKIE\_PAIR CookiePair;
11. PFW\_AUTH\_INFO pFirstAuth;
12. PFW\_AUTH\_INFO pSecondAuth;
13. unsigned long dwP1SaFlags;
14. } FW\_PHASE1\_SA\_DETAILS,
15. \*PFW\_PHASE1\_SA\_DETAILS;

**SaId:**  A 64-bit integer that uniquely identifies the security association.

**KeyModuleType:**  The keying protocol used, [**IKE**](#gt_294fef97-5790-4d41-971e-dd255b783e68) or [**AuthIP**](#gt_3791f3e1-cf2f-4605-9fcc-54f526f036cf). The field MUST contain only a value from the [FW\_PHASE1\_KEY\_MODULE\_TYPE](#Section_f0944fb9dc8741128b5fe897027f2b73) enumeration.

**Endpoints:**  This field contains IP address information of the two [**endpoints**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee) that established this security association. An address of zero means the security association applies to any endpoint.

**SelectedProposal:**  This is the Phase1 cryptographic suite that was selected by the negotiation of the keying protocol.

**dwProposalLifetimeKBytes:**  Currently not supported.

**dwProposalLifetimeMinutes:**  This field specifies the lifetime in minutes of this security association before a rekey MUST happen.

**dwProposalMaxNumPhase2:**  This field specifies the number of Phase2 (quick mode) negotiations (rekeys) that can happen before this security association MUST be renegotiated.

**CookiePair:**  This value is used for diagnostics.

**pFirstAuth:**  A pointer to an [FW\_AUTH\_INFO](#Section_cd08ebed295d4c7786c64ba87aa4df00) structure that contains the information that resulted from the method negotiated and used for first authentication. This pointer MUST NOT be null.

**pSecondAuth:**  A pointer to an FW\_AUTH\_INFO structure that contains the information that resulted from the method negotiated and used for second authentication. If the field is NULL, the second authentication was not performed.

**dwP1SaFlags:**  Reserved value and not currently used. It MUST be set to 0.

### FW\_PHASE2\_TRAFFIC\_TYPE

This enumeration identifies the two types of traffic enforcement modes that [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) supports. It is defined in the [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) for future use.

1. typedef enum \_tag\_FW\_PHASE2\_TRAFFIC\_TYPE
2. {
3. FW\_PHASE2\_TRAFFIC\_TYPE\_INVALID = 0,
4. FW\_PHASE2\_TRAFFIC\_TYPE\_TRANSPORT = 1,
5. FW\_PHASE2\_TRAFFIC\_TYPE\_TUNNEL = 2,
6. FW\_PHASE2\_TRAFFIC\_TYPE\_MAX = 3
7. } FW\_PHASE2\_TRAFFIC\_TYPE;

**FW\_PHASE2\_TRAFFIC\_TYPE\_INVALID:** This value MUST NOT be used. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 0.

**FW\_PHASE2\_TRAFFIC\_TYPE\_TRANSPORT:** This value represents IPsec transport mode, which happens directly between two [**endpoints**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee). This symbolic constant has a value of 1.

**FW\_PHASE2\_TRAFFIC\_TYPE\_TUNNEL:** This value represents IPsec tunnel mode, which uses two other endpoints to tunnel through them when the original endpoints communicate. This symbolic constant has a value of 2.

**FW\_PHASE2\_TRAFFIC\_TYPE\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 3.

### FW\_PHASE2\_SA\_DETAILS

This structure represents a [**security association**](#gt_67cbf867-7a49-41f3-a68f-37b5f9035acb) that is established after the quick mode negotiations take place; it contains the selected algorithms to enforce [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb).

1. typedef struct \_tag\_FW\_PHASE2\_SA\_DETAILS {
2. unsigned \_\_int64 SaId;
3. [range(FW\_DIR\_INVALID+1,FW\_DIR\_MAX-1)]
4. FW\_DIRECTION Direction;
5. FW\_ENDPOINTS Endpoints;
6. unsigned short wLocalPort;
7. unsigned short wRemotePort;
8. unsigned short wIpProtocol;
9. FW\_PHASE2\_CRYPTO\_SUITE SelectedProposal;
10. FW\_PHASE2\_CRYPTO\_PFS Pfs;
11. GUID TransportFilterId;
12. unsigned long dwP2SaFlags;
13. } FW\_PHASE2\_SA\_DETAILS,
14. \*PFW\_PHASE2\_SA\_DETAILS;

**SaId:**  A 64-bit integer number that uniquely identifies the security association.

**Direction:**  This field specifies the direction of the traffic this security association is securing.

**Endpoints:**  This field contains IP address information of the two [**endpoints**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee) that established this security association. An address of zero means the security association applies to any endpoint.

**wLocalPort:**  This field specifies the port of the local endpoint that is used in the traffic secured by this security association. A value of 0 specifies any port.

**wRemotePort:**  This field specifies the port of the remote endpoint that is used in the traffic secured by this security association. A value of 0 specifies any port.

**wIpProtocol:**  This field specifies the protocol of the traffic secured by this security association. If the value is within the range 0 to 255, the value describes a protocol as in IETF IANA numbers (for more information, see [[IANA-PROTO-NUM]](https://go.microsoft.com/fwlink/?LinkId=89889)). If the value is 256, the rule matches ANY protocol.

**SelectedProposal:**  This field contains the Phase2 cryptographic suite selected by the negotiation that is used by this security association to enforce IPsec.

**Pfs:**  This field specifies the [**perfect forward secrecy**](#gt_5d8948bc-5e32-483b-906d-42f785d0df18) used by this security association.

**TransportFilterId:**  This [**GUID**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1) MAY contain additional implementation-specific[<18>](#Appendix_A_18" \o "Product behavior note 18) information about the security association. The client MUST ignore this value.

**dwP2SaFlags:**  Reserved value and not currently used. It MUST be set to 0.

### FW\_PROFILE\_CONFIG\_VALUE

This union defines the value stored by each of the different policy configuration values identified by the enumeration [FW\_PROFILE\_CONFIG](#Section_5a6e0d39802d456bb483c7360566fcdd). This data type is used to pass different types of values across the same structure on function calls.

1. typedef
2. [switch\_type(FW\_PROFILE\_CONFIG)]
3. union \_FW\_PROFILE\_CONFIG\_VALUE {
4. [case(FW\_PROFILE\_CONFIG\_LOG\_FILE\_PATH)]
5. [string, range(1,10001)] wchar\_t\* wszStr;
6. [case(FW\_PROFILE\_CONFIG\_DISABLED\_INTERFACES)]
7. PFW\_INTERFACE\_LUIDS pDisabledInterfaces;
8. [case(FW\_PROFILE\_CONFIG\_ENABLE\_FW, FW\_PROFILE\_CONFIG\_DISABLE\_STEALTH\_MODE, FW\_PROFILE\_CONFIG\_SHIELDED, FW\_PROFILE\_CONFIG\_DISABLE\_UNICAST\_RESPONSES\_TO\_MULTICAST\_BROADCAST, FW\_PROFILE\_CONFIG\_LOG\_DROPPED\_PACKETS, FW\_PROFILE\_CONFIG\_LOG\_SUCCESS\_CONNECTIONS, FW\_PROFILE\_CONFIG\_LOG\_IGNORED\_RULES, FW\_PROFILE\_CONFIG\_LOG\_MAX\_FILE\_SIZE, FW\_PROFILE\_CONFIG\_DISABLE\_INBOUND\_NOTIFICATIONS, FW\_PROFILE\_CONFIG\_AUTH\_APPS\_ALLOW\_USER\_PREF\_MERGE, FW\_PROFILE\_CONFIG\_GLOBAL\_PORTS\_ALLOW\_USER\_PREF\_MERGE, FW\_PROFILE\_CONFIG\_ALLOW\_LOCAL\_POLICY\_MERGE, FW\_PROFILE\_CONFIG\_ALLOW\_LOCAL\_IPSEC\_POLICY\_MERGE, FW\_PROFILE\_CONFIG\_DEFAULT\_OUTBOUND\_ACTION, FW\_PROFILE\_CONFIG\_DEFAULT\_INBOUND\_ACTION, FW\_PROFILE\_CONFIG\_DISABLE\_STEALTH\_MODE\_IPSEC\_SECURED\_PACKET\_EXEMPTION )]
9. unsigned long\* pdwVal;
10. } FW\_PROFILE\_CONFIG\_VALUE,
11. \*PFW\_PROFILE\_CONFIG\_VALUE;

**wszStr:**  This field contains a pointer to a [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string. It is used when the data type of the configuration value is a string.

**pDisabledInterfaces:**  This field contains a pointer to an [FW\_INTERFACE\_LUIDS](#Section_ea420d0f03ed48e5b786621db56419d5) data type, which holds a list of [**GUIDs**](#gt_f49694cc-c350-462d-ab8e-816f0103c6c1). This field is custom marshaled, so it is passed as a plain buffer. The following diagrams show how the structures are marshaled.

On 32-bit servers:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| dwNumLUIDs | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| pLUIDs | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| GUID1 (16 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

On 64-bit servers:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| dwNumLUIDs | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| pLUIDs | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| GUID1 (16 bytes) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ... | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**pdwVal:**  This field contains a pointer to an **unsigned long**. It is used when the data type of the configuration value is an **unsigned long**.

### FW\_MM\_RULE

This structure is used to represent a main mode rule.

1. typedef struct \_tag\_FW\_MM\_RULE {
2. struct \_tag\_FW\_MM\_RULE\* pNext;
3. unsigned SHORT wSchemaVersion;
4. [string, range(1,512), ref] wchar\_t\* wszRuleId;
5. [string, range(1,10001)] wchar\_t\* wszName;
6. [string, range(1,10001)] wchar\_t\* wszDescription;
7. unsigned LONG dwProfiles;
8. FW\_ADDRESSES Endpoint1;
9. FW\_ADDRESSES Endpoint2;
10. [string, range(1,255)] wchar\_t\* wszPhase1AuthSet;
11. [string, range(1,255)] wchar\_t\* wszPhase1CryptoSet;
12. unsigned SHORT wFlags;
13. [string, range(1,10001)] wchar\_t wszEmbeddedContext;
14. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
15. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX-1)]
16. FW\_RULE\_ORIGIN\_TYPE Origin;
17. [string, range(1,10001)] wchar\_t wszGPOName;
18. FW\_RULE\_STATUS Status;
19. signed LONG Reserved;
20. [size\_is((Reserved & FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA) ? 1 : 0)]
21. FW\_OBJECT\_METADATA pMetaData;
22. } FW\_MM\_RULE,
23. \*PFW\_MM\_RULE;

**pNext:**  A pointer to the next FW\_MM\_RULE in the list.

**wSchemaVersion:**  Specifies the version of the rule.

**wszRuleId:**  A pointer to a [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string that uniquely identifies the rule.

**wszName:**  A pointer to a Unicode string that provides a friendly name for the rule.

**wszDescription:**  A pointer to a Unicode string that provides a friendly description for the rule.

**dwProfiles:**  A bitmask of the [FW\_PROFILE\_TYPE](#Section_7704e238174d4a5eb8095f3787dd8acc) flags. It is a condition that matches traffic on the specified profiles.

**Endpoint1:**  A condition that specifies the addresses of the first host of the traffic that the rule matches. An empty EndPoint1 structure means this condition is not applied (no match).

**Endpoint2:**  A condition that specifies the addresses of the second host of the traffic that the rule matches. An empty EndPoint2 structure means this condition is not applied (no match).

**wszPhase1AuthSet:**  A Unicode string that represents the set identifier of a Phase1 authentication sets policy objects.

**wFlags:**  Bit flags from [FW\_CS\_RULE\_FLAGS](#Section_567fdc537b1c418b8b11dd7267f75bad).

**wszEmbeddedContext:**  A pointer to a Unicode string that specifies a group name for this rule. Other components in the system use this string to enable or disable a group of rules by verifying that all rules have the same group name.

**PlatformValidityList:**  A condition in a rule that determines whether or not the rule is enforced by the local computer based on the local computer's platform information. The rule is enforced only if the local computer's operating system platform is an element of the set described by **PlatformValidityList**.[<19>](#Appendix_A_19" \o "Product behavior note 19)

**Origin:**  This field is the rule origin, as specified in the [FW\_RULE\_ORIGIN\_TYPE](#Section_9d295321d75c41c0ab0d7a78df40f77c) enumeration. It MUST be filled on enumerated rules and ignored on input.

**wszGPOName:**  A pointer to a Unicode string containing the displayName of the [**GPO**](#gt_dec32233-8776-4151-91a0-8624a2b9abb0) containing this object. When adding a new object, this field is not used. The client SHOULD set the value to NULL, and the server MUST ignore the value. When enumerating an existing object, if the client does not set the FW\_ENUM\_RULES\_FLAG\_RESOLVE\_GPO\_NAME flag, the server MUST set the value to NULL. Otherwise, the server MUST set the value to the displayName of the GPO containing the object or NULL if the object is not contained within a GPO. For details about how the server initializes an object from a GPO, see section [3.1.3](#Section_e8924ac5aa4a41d1bf654f46b3d399aa). For details about how the displayName of a GPO is stored, see [[MS-GPOL]](%5bMS-GPOL%5d.pdf#Section_62d1292462524052996f161d2b9019f4) section 2.3.

**Status:**  The status code of the rule, as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out when the structure is returned as output. On input, this field MUST be set to FW\_RULE\_STATUS\_OK.

**Reserved:**  This member is not used, other than to instruct [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331), by using the FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA flag, that a pointer to an [FW\_OBJECT\_METADATA](#Section_cecfb58e2c794b7db92c7666fa06d559) structure is present. It has no semantic meaning to the object itself.

**pMetaData:**  A pointer to an FW\_OBJECT\_METADATA structure that contains specific metadata about the current state of the connection security rule.

### FW\_CONN\_HANDLE

This type contains an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle, as specified in [[C706]](https://go.microsoft.com/fwlink/?LinkId=89824) section 2, to an RPC interface that implements the Firewall and Advanced Security Protocol. For information on handle\_t, see [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.1.3.

This type is declared as follows:

1. typedef handle\_t FW\_CONN\_HANDLE;

### FW\_MATCH\_KEY

This enumeration describes the keys that a query is allowed to match.

1. typedef enum \_tag\_FW\_MATCH\_KEY
2. {
3. FW\_MATCH\_KEY\_PROFILE = 0,
4. FW\_MATCH\_KEY\_STATUS = 1,
5. FW\_MATCH\_KEY\_OBJECTID = 2,
6. FW\_MATCH\_KEY\_FILTERID = 3,
7. FW\_MATCH\_KEY\_APP\_PATH = 4,
8. FW\_MATCH\_KEY\_PROTOCOL = 5,
9. FW\_MATCH\_KEY\_LOCAL\_PORT = 6,
10. FW\_MATCH\_KEY\_REMOTE\_PORT = 7,
11. FW\_MATCH\_KEY\_GROUP = 8,
12. FW\_MATCH\_KEY\_SVC\_NAME = 9,
13. FW\_MATCH\_KEY\_DIRECTION = 10,
14. FW\_MATCH\_KEY\_LOCAL\_USER\_OWNER = 11,
15. FW\_MATCH\_KEY\_PACKAGE\_ID = 12,
16. FW\_MATCH\_KEY\_FQBN = 13,
17. FW\_MATCH\_KEY\_COMPARTMENT\_ID = 14,
18. FW\_MATCH\_KEY\_MAX = 15
19. } FW\_MATCH\_KEY;

**FW\_MATCH\_KEY\_PROFILE:** This key matches the profile conditions of the queried object. This symbolic constant has a value of 0.

**FW\_MATCH\_KEY\_STATUS:** This key matches the status conditions of the queried object. This symbolic constant has a value of 1.

**FW\_MATCH\_KEY\_OBJECTID:** This key matches the object ID (rule ID or set ID) of the queried object. This symbolic constant has a value of 2.

**FW\_MATCH\_KEY\_FILTERID:** This value is not used on the wire. This symbolic constant has a value of 3.

**FW\_MATCH\_KEY\_APP\_PATH:** This key matches the application condition of the queried object. This symbolic constant has a value of 4.

**FW\_MATCH\_KEY\_PROTOCOL:** This key matches the protocol condition of the queried object. This symbolic constant has a value of 5.

**FW\_MATCH\_KEY\_LOCAL\_PORT:** This key matches the [**TCP**](#gt_b08d36f6-b5c6-4ce4-8d2d-6f2ab75ea4cb) or UDP local port condition of the queried object. This symbolic constant has a value of 6.

**FW\_MATCH\_KEY\_REMOTE\_PORT:** This key matches the TCP or UDP remote port condition of the queried object. This symbolic constant has a value of 7.

**FW\_MATCH\_KEY\_GROUP:** This key matches the group name (the Embedded context field) of the queried object. This symbolic constant has a value of 8.

**FW\_MATCH\_KEY\_SVC\_NAME:** This key matches the service name condition of the queried object. This symbolic constant has a value of 9.

**FW\_MATCH\_KEY\_DIRECTION:** This key matches the direction condition of the queried object. This symbolic constant has a value of 10.

**FW\_MATCH\_KEY\_LOCAL\_USER\_OWNER:** This key matches the local user owner condition of the queried object. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used. This symbolic constant has a value of 11.

**FW\_MATCH\_KEY\_PACKAGE\_ID:** This key matches the package ID condition of the queried object. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used. This symbolic constant has a value of 12.

**FW\_MATCH\_KEY\_FQBN:** This key matches the [**fully qualified binary name (FQBN)**](#gt_bbf47ea1-11e7-447c-848d-5a1277648312) condition of the queried object. For schema versions 0x0200 through 0x021A, this value is invalid and MUST NOT be used. This symbolic constant has a value of 13.

**FW\_MATCH\_KEY\_COMPARTMENT\_ID:** This key matches the compartment ID condition of the queried object. For schema versions 0x0200 through 0x021A, this value is invalid and MUST NOT be used. This symbolic constant has a value of 14.

**FW\_MATCH\_KEY\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 15.

### FW\_DATA\_TYPE

This enumeration describes the data types that this protocol uses in generic structures. It is currently used only in section [2.2.88](#Section_3af7dd01b8c648b0b179824a10464911).

1. typedef enum \_tag\_FW\_DATA\_TYPE
2. {
3. FW\_DATA\_TYPE\_EMPTY,
4. FW\_DATA\_TYPE\_UINT8,
5. FW\_DATA\_TYPE\_UINT16,
6. FW\_DATA\_TYPE\_UINT32,
7. FW\_DATA\_TYPE\_UINT64,
8. FW\_DATA\_TYPE\_UNICODE\_STRING
9. } FW\_DATA\_TYPE;

**FW\_DATA\_TYPE\_EMPTY:** The value SHOULD be empty and not used. This symbolic constant has a value of zero.

**FW\_DATA\_TYPE\_UINT8:** This data type is a UINT8, which is an 8-bit unsigned integer. This symbolic constant has a value of 1.

**FW\_DATA\_TYPE\_UINT16:** This data type is a UINT16, which is a 16-bit unsigned integer. This symbolic constant has a value of 2.

**FW\_DATA\_TYPE\_UINT32:** This data type is a UINT32, which is a 32-bit unsigned integer. This symbolic constant has a value of 3.

**FW\_DATA\_TYPE\_UINT64:** This data type is a UINT64, which is a 64-bit unsigned integer. This symbolic constant has a value of 4.

**FW\_DATA\_TYPE\_UNICODE\_STRING:** This data type is a [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string. This symbolic constant has a value of 5.

### FW\_MATCH\_VALUE

This structure is used to generically store different data types.

1. typedef struct \_tag\_FW\_MATCH\_VALUE {
2. FW\_DATA\_TYPE type;
3. [switch\_type(FW\_DATA\_TYPE), switch\_is(type)]
4. union {
5. [case(FW\_DATA\_TYPE\_UINT8)]
6. unsigned CHAR uInt8;
7. [case(FW\_DATA\_TYPE\_UINT16)]
8. unsigned SHORT uInt16;
9. [case(FW\_DATA\_TYPE\_UINT32)]
10. unsigned LONG uInt32;
11. [case(FW\_DATA\_TYPE\_UINT64)]
12. unsigned \_\_int64 uInt64;
13. [case(FW\_DATA\_TYPE\_UNICODE\_STRING)]
14. struct {
15. [string, range(1,10001)] wchar\_t\* wszString;
16. };
17. [case(FW\_DATA\_TYPE\_EMPTY)]
18. ;
19. };
20. } FW\_MATCH\_VALUE;

**type:**  This field identifies the data type that is stored in the structure.

**uInt8:**  This field contains an 8-bit unsigned integer.

**uInt16:**  This field contains a 16-bit unsigned integer.

**uInt32:**  This field contains a 32-bit unsigned integer.

**uInt64:**  This field contains a 64-bit unsigned integer.

**wszString:**  This field contains a pointer to a [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) string.

### FW\_MATCH\_TYPE

This enumeration specifies how a match key is matched against an object.

1. typedef enum \_tag\_FW\_MATCH\_TYPE
2. {
3. FW\_MATCH\_TYPE\_TRAFFIC\_MATCH = 0,
4. FW\_MATCH\_TYPE\_EQUAL = 1,
5. FW\_MATCH\_TYPE\_MAX = 2
6. } FW\_MATCH\_TYPE;

**FW\_MATCH\_TYPE\_TRAFFIC\_MATCH:** The match operation evaluates to TRUE for all objects that match the network traffic that is represented by the value matched against. This symbolic constant has a value of 0.

**FW\_MATCH\_TYPE\_EQUAL:** The match operation evaluates to TRUE for all objects that have a value equal to the one matched against. This symbolic constant has a value of 1.

**FW\_MATCH\_TYPE\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 2.

### FW\_QUERY\_CONDITION

This structure specifies a condition of a query. A condition can evaluate to TRUE or FALSE. It contains a match key that identifies what to match, a match value that identifies what to match with, and a match type that identifies how to match.

1. typedef struct \_tag\_FW\_QUERY\_CONDITION {
2. FW\_MATCH\_KEY matchKey;
3. FW\_MATCH\_TYPE matchType;
4. FW\_MATCH\_VALUE matchValue;
5. } FW\_QUERY\_CONDITION,
6. \*PFW\_QUERY\_CONDITION;

**matchKey:**  This field identifies what information to match.

**matchType:**  This field identifies how to perform the match operation.

**matchValue:**  This field identifies what to match with.

A query condition structure MUST pass the following semantics checks:

* The **matchKey** field MUST have a valid **FW\_MATCH\_KEY** value that is less than FW\_MATCH\_KEY\_MAX, MUST be a string of 1 or more characters, and MUST NOT be greater than or equal to 255 characters.
* The **matchType** field MUST have a valid **FW\_MATCH\_TYPE** value that is less than FW\_MATCH\_KEY\_MAX.
* If the **matchType** field is equal to FW\_MATH\_TYPE\_EQUAL, the **matchKey** field MUST be either FW\_MATCH\_KEY\_GROUP or FW\_MATCH\_KEY\_DIRECTION.
* If the **matchKey** field is equal to FW\_MATCH\_KEY\_PROFILE or FW\_MATCH\_KEY\_STATUS, the **matchValue** MUST have its type field equal to FW\_DATA\_TYPE\_UINT32.
* If the **matchKey** field is equal to FW\_MATCH\_KEY\_FILTERID, the **matchValue** MUST have its type field equal to FW\_DATA\_TYPE\_UINT64.
* If the **matchKey** field is equal to FW\_MATCH\_KEY\_PROTOCOL, FW\_MATCH\_KEY\_LOCAL\_PORT, or FW\_MATCH\_KEY\_REMOTE\_PORT; then the **matchValue** MUST have its type field equal to FW\_DATA\_TYPE\_UINT16.
* If the **matchKey** field is equal to FW\_MATCH\_KEY\_OBJECTID, FW\_MATCH\_KEY\_APP\_PATH, FW\_MATCH\_KEY\_GROUP, or FW\_MATCH\_KEY\_SVC\_NAME; then the **matchValue** MUST have its type field equal to FW\_DATA\_TYPE\_UNICODE\_STRING.

### FW\_QUERY\_CONDITIONS

This structure is used to contain a number of [FW\_QUERY\_CONDITION](#Section_fa19141ffc3c45159555b6db62c475aa) elements. This structure can evaluate to either TRUE or FALSE. It evaluates to TRUE if all query condition elements evaluate to TRUE; otherwise, it evaluates to FALSE.

1. typedef struct \_tag\_FW\_QUERY\_CONDITIONS {
2. unsigned LONG dwNumEntries;
3. [size\_is(dwNumEntries)] FW\_QUERY\_CONDITION\* pAndedConditions;
4. } FW\_QUERY\_CONDITIONS,
5. \*PFW\_QUERY\_CONDITIONS;

**dwNumEntries:**  Specifies the number of query conditions that the structure contains.

**pAndedConditions:**  A pointer to an array of FW\_QUERY\_CONDITIONS elements, which are all logically AND'd together. The number of elements is given by dwNumEntries.

A query condition structure MUST pass the following semantic checks:

* If the **dwNumEntries** field is zero, the **AndedConditions** field MUST be NULL; and if the **dwNumEntries** field is not zero, the **AndedConditions** field MUST NOT be NULL.
* If the **AndedConditions** field array has a FW\_QUERY\_CONDITION element with the **matchKey** field equal to FW\_MATCH\_KEY\_LOCAL\_PORT or FW\_MATCH\_KEY\_REMOTE\_PORT at position N of the array, the array MUST have another element whose **matchKey** field is equal to FW\_MATCH\_KEY\_PROTOCOL at position M, where M < N.
* All elements of the **AndedConditions** array MUST have valid FW\_QUERY\_CONDITION structures.

### FW\_QUERY

This structure is used to query objects from the store. The structure contains a number of [FW\_QUERY\_CONDITIONS](#Section_98765a8c9a0c471bb2ee56c931ea5604) elements. This structure can evaluate to either TRUE or FALSE. It evaluates to TRUE if at least one of the query conditions containers evaluates to TRUE; otherwise, if all evaluate to FALSE, it evaluates to FALSE.

1. typedef struct \_tag\_FW\_QUERY {
2. unsigned SHORT wSchemaVersion;
3. unsigned LONG dwNumEntries;
4. [size\_is(dwNumEntries)] FW\_QUERY\_CONDITIONS\* ORConditions;
5. FW\_RULE\_STATUS Status;
6. } FW\_QUERY,
7. \*PFW\_QUERY;

**wSchemaVersion:**  The schema version of the query object. The version MUST be at least 0x00020A.

**dwNumEntries:**  This field specifies the number of query conditions containers that the structure contains.

**ORConditions:**  A pointer to an array of FW\_QUERY\_CONDITIONS elements, which are all logically OR'd together. The number of elements is given by **dwNumEntries**.

**Status:**  The status code of the query, as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out when the structure is returned as output. On input, this field SHOULD be set to FW\_RULE\_STATUS\_OK.

The following are semantic checks that query object MUST pass:

* The **wSchemaVersion** MUST NOT be less than 0x00020A.
* If the **dwNumEntries** field is zero, the **ORConditions** field MUST be NULL, and if the **dwNumEntries** field is not zero, the **ORConditions** field MUST NOT be NULL.
* The **ORConditions** field MUST have valid FW\_QUERY\_CONDITIONS elements.
* If the query object is used for querying connection security rules, it MUST NOT have any conditions with **matchKey** equal to FW\_MATCH\_KEY\_APP\_PATH or FW\_MATCH\_KEY\_SVC\_NAME.
* If the query object is being used for querying main mode rules, it MUST NOT have any conditions with **matchKey** equal to FW\_MATCH\_KEY\_PROTOCOL, FW\_MATCH\_KEY\_LOCAL\_PORT, FW\_MATCH\_KEY\_REMOTE\_PORT, FW\_MATCH\_KEY\_GROUP, or FW\_MATCH\_KEY\_DIRECTION, or any of the match keys disallowed by connection security rules.
* If the query object is being used for querying authentication or cryptographic sets, it MUST NOT have any conditions with **matchKey** equal to FW\_MATCH\_KEY\_PROFILE or FW\_MATCH\_KEY\_FILTERID, or any of the match keys disallowed by main mode rules.

### FW\_POLICY\_STORE\_HANDLE

1. typedef [context\_handle] void\* FW\_POLICY\_STORE\_HANDLE;
2. typedef [ref] FW\_POLICY\_STORE\_HANDLE\* PFW\_POLICY\_STORE\_HANDLE;

This type is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle. It is a handle to a policy store exposed by this protocol. This handle is used to manage the policy contained in each store. Policy stores are identified by the [FW\_STORE\_TYPE](#Section_37ebed958abf472c8b4b7a510a2a6baa) enumeration.

### FW\_PRODUCT\_HANDLE

This type is declared as follows:

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

This type is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) context handle. It is a handle to the third-party software components that are registered with the firewall and advanced security component which are exposed through this protocol.

### FW\_KEY\_MODULE

This enumeration defines the possible keying modules that the policy rule applies to.

1. typedef enum
2. {
3. FW\_KEY\_MODULE\_DEFAULT = 0,
4. FW\_KEY\_MODULE\_IKEv1 = 1,
5. FW\_KEY\_MODULE\_AUTHIP = 2,
6. FW\_KEY\_MODULE\_IKEv2 = 3,
7. FW\_KEY\_MODULE\_MAX = 4
8. } FW\_KEY\_MODULE;

**FW\_KEY\_MODULE\_DEFAULT:** This value represents the default keying modules. The default keying modules are implementation-specific.[<20>](#Appendix_A_20" \o "Product behavior note 20)

**FW\_KEY\_MODULE\_IKEv1:** This value represents a keying module implementing the [**Internet Key Exchange (IKE)**](#gt_294fef97-5790-4d41-971e-dd255b783e68) protocol as specified in [[RFC2409]](https://go.microsoft.com/fwlink/?LinkId=90349).

**FW\_KEY\_MODULE\_AUTHIP:** This value represents a keying module implementing the Authenticated Internet protocol as specified in [[MS-AIPS]](%5bMS-AIPS%5d.pdf#Section_eee3de6438474451978e9513ff187d30).

**FW\_KEY\_MODULE\_IKEv2:** This value represents a keying module implementing the [**Internet Key Exchange (IKEv2)**](#gt_2cd83a7f-fc17-4283-b3f3-59feb25114bf) protocol as specified in [[RFC4306]](https://go.microsoft.com/fwlink/?LinkId=90469).

**FW\_KEY\_MODULE\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined to provide for simplicity when writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code. This symbolic constant has a value of 4.

### FW\_TRUST\_TUPLE\_KEYWORD

This enumeration represents flags that are used to identify trust tuples. The traffic corresponding to these keywords changes dynamically and is tracked by the TrustTuples object (section [3.1.1](#Section_43507d538955416db913dfb27dc76b17)). All the flags supported by a given schema version can be combined.

1. typedef enum \_tag\_FW\_TRUST\_TUPLE\_KEYWORD\_NONE
2. {
3. FW\_TRUST\_TUPLE\_KEYWORD\_NONE = 0x0000,
4. FW\_TRUST\_TUPLE\_KEYWORD\_PROXIMITY = 0x0001,
5. FW\_TRUST\_TUPLE\_KEYWORD\_PROXIMITY\_SHARING = 0x0002,
6. FW\_TRUST\_TUPLE\_KEYWORD\_WFD\_PRINT = 0x0004,
7. FW\_TRUST\_TUPLE\_KEYWORD\_WFD\_DISPLAY = 0x0008,
8. FW\_TRUST\_TUPLE\_KEYWORD\_WFD\_DEVICES = 0x0010,
9. FW\_TRUST\_TUPLE\_KEYWORD\_WFD\_KM\_DRIVER = 0x0020,
10. FW\_TRUST\_TUPLE\_KEYWORD\_UPNP = 0x0040,
11. FW\_TRUST\_TUPLE\_KEYWORD\_MAX = 0x0080,
12. FW\_TRUST\_TUPLE\_KEYWORD\_MAX\_V2\_20 = 0x0004,
13. FW\_TRUST\_TUPLE\_KEYWORD\_MAX\_V2\_26 = 0x0020
14. } FW\_TRUST\_TUPLE\_KEYWORD\_NONE;

**FW\_TRUST\_TUPLE\_KEYWORD\_NONE:** This value means that none of the following flags are set. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code.

**FW\_TRUST\_TUPLE\_KEYWORD\_PROXIMITY:** Represents all traffic matching a trust tuple in the**TrustTuples** collection where **IsProximity** is true.

**FW\_TRUST\_TUPLE\_KEYWORD\_PROXIMITY\_SHARING:** Represents all traffic matching a trust tuple in the **TrustTuples** collection where **IsProximitySharing** is true.

**FW\_TRUST\_TUPLE\_KEYWORD\_WFD\_PRINT:** Represents all traffic matching a trust tuple in the **TrustTuples** collection where **IsWFDPrint** is true.

**FW\_TRUST\_TUPLE\_KEYWORD\_WFD\_DISPLAY:** Represents all traffic matching a trust tuple in the **TrustTuples** collection where **IsWFDDevices** is true.

**FW\_TRUST\_TUPLE\_KEYWORD\_WFD\_DEVICES:** Represents all traffic matching a trust tuple in the **TrustTuples** collection where **IsWFDDevices** is true.

**FW\_TRUST\_TUPLE\_KEYWORD\_WFD\_KM\_DRIVER:** Represents all traffic matching a trust tuple in the **TrustTuples** collection, where **IsWFDMaUsbWirelessDocking** is true.

**FW\_TRUST\_TUPLE\_KEYWORD\_UPNP:** Represents all traffic that matches the UPnP tuple for Secure Sockets interaction with Teredo.

**FW\_TRUST\_TUPLE\_KEYWORD\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 80.[<21>](#Appendix_A_21" \o "Product behavior note 21)

**FW\_TRUST\_TUPLE\_KEYWORD\_MAX\_V2\_20:** This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x0214 and earlier. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 4.

**FW\_TRUST\_TUPLE\_KEYWORD\_MAX\_V2\_26:** This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x021A and earlier. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 20.

### FW\_RULE2\_10

This structure represents a firewall rule that is used by the 2.10 binary version servers and clients (see sections [1.7](#Section_6b14b8f991ab4f09873f9e2334196cdc) and [2.2.41](#Section_faf4ffbe1d5140adae902230f2c0b6a9)). The fields of this structure are identical to the FW\_RULE structure and its meanings are covered in section [2.2.36](#Section_8c008258166d46d49090f2ffaa01be4b).

1. typedef struct \_tag\_FW\_RULE2\_10 {
2. struct \_tag\_FW\_RULE2\_10\* pNext;
3. unsigned short wSchemaVersion;
4. [string, range(1, 10001), ref] wchar\_t\* wszRuleId;
5. [string, range(1, 10001)] wchar\_t\* wszName;
6. [string, range(1, 10001)] wchar\_t\* wszDescription;
7. unsigned long dwProfiles;
8. [range(FW\_DIR\_INVALID, FW\_DIR\_OUT)]
9. FW\_DIRECTION Direction;
10. [range(0, 256)] unsigned short wIpProtocol;
11. [switch\_type(unsigned short), switch\_is(wIpProtocol)]
12. union {
13. [case(6,17)]
14. struct {
15. FW\_PORTS LocalPorts;
16. FW\_PORTS RemotePorts;
17. };
18. [case(1)]
19. FW\_ICMP\_TYPE\_CODE\_LIST V4TypeCodeList;
20. [case(58)]
21. FW\_ICMP\_TYPE\_CODE\_LIST V6TypeCodeList;
22. [default] ;
23. };
24. FW\_ADDRESSES LocalAddresses;
25. FW\_ADDRESSES RemoteAddresses;
26. FW\_INTERFACE\_LUIDS LocalInterfaceIds;
27. unsigned long dwLocalInterfaceTypes;
28. [string, range(1, 10001)] wchar\_t\* wszLocalApplication;
29. [string, range(1, 10001)] wchar\_t\* wszLocalService;
30. [range(FW\_RULE\_ACTION\_INVALID, FW\_RULE\_ACTION\_MAX)]
31. FW\_RULE\_ACTION Action;
32. unsigned short wFlags;
33. [string, range(1, 10001)] wchar\_t\* wszRemoteMachineAuthorizationList;
34. [string, range(1, 10001)] wchar\_t\* wszRemoteUserAuthorizationList;
35. [string, range(1, 10001)] wchar\_t\* wszEmbeddedContext;
36. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
37. FW\_RULE\_STATUS Status;
38. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX)]
39. FW\_RULE\_ORIGIN\_TYPE Origin;
40. [string, range(1, 10001)] wchar\_t\* wszGPOName;
41. unsigned long Reserved;
42. [size\_is((Reserved & FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA) ? 1 : 0)]
43. PFW\_OBJECT\_METADATA pMetaData;
44. } FW\_RULE2\_10,
45. \*PFW\_RULE2\_10;

### FW\_AUTH\_SET\_FLAGS

This enumeration represents flags that can be specified in authentication sets of section [2.2.64](#Section_bbec7ab70a1d49a49c8ea4784bbe6f16).

1. typedef enum \_tag\_FW\_AUTH\_SET\_FLAGS
2. {
3. FW\_AUTH\_SET\_FLAGS\_NONE = 0x00,
4. FW\_AUTH\_SET\_FLAGS\_EMPTY = 0x01,
5. FW\_AUTH\_SET\_FLAGS\_MAX = 0x02,
6. FW\_AUTH\_SET\_FLAGS\_MAX\_2\_10 = 0x01
7. } FW\_AUTH\_SET\_FLAGS;

**FW\_AUTH\_SET\_FLAGS\_NONE:** This value means that none of the following flags are set. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code.

**FW\_AUTH\_SET\_FLAGS\_EMPTY:** If this flag is set, the authentication set does not contain any authentication suites. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used.

**FW\_AUTH\_SET\_FLAGS\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 2.

**FW\_AUTH\_SET\_FLAGS\_MAX\_2\_10:** This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x020A and earlier. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 1.

### FW\_CRYPTO\_SET\_FLAGS

This enumeration represents flags that can be specified in crypto sets of section [2.2.73](#Section_a468fe9e113b4155a63d0db3aac12619).

1. typedef enum \_tag\_FW\_CRYPTO\_SET\_FLAGS
2. {
3. FW\_CRYPTO\_SET\_FLAGS\_NONE = 0x00,
4. FW\_CRYPTO\_SET\_FLAGS\_EMPTY = 0x01,
5. FW\_CRYPTO\_SET\_FLAGS\_MAX = 0x02,
6. FW\_CRYPTO\_SET\_FLAGS\_MAX\_2\_10 = 0x01
7. } FW\_CRYPTO\_SET\_FLAGS;

**FW\_CRYPTO\_SET\_FLAGS\_NONE:** This value means that none of the following flags are set. It is defined for simplicity in writing [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) definitions and code.

**FW\_CRYPTO\_SET\_FLAGS\_EMPTY:** If this flag is set, the crypto set does not contain any crypto suites. For schema versions 0x0200, 0x0201, and 0x020A, this value is invalid and MUST NOT be used.

**FW\_CRYPTO\_SET\_FLAGS\_MAX:** This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 2.

**FW\_CRYPTO\_SET\_FLAGS\_MAX\_2\_10:** This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x020A and earlier. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 1.

### FW\_NETWORK\_NAMES

The FW\_NETWORK\_NAMES structure represents a firewall rule that is used by the 2.24 binary version servers and clients (see sections [1.7](#Section_6B14B8F991AB4F09873F9E2334196CDC) and [2.2.41](#Section_FAF4FFBE1D5140ADAE902230F2C0B6A9)).

1. typedef struct \_tag\_FW\_NETWORK\_NAMES {
2. DWORD dwNumEntries;
3. [string, unique, size\_is(dwNumEntries,)]
4. LPWSTR\* wszNames;
5. } FW\_NETWORK\_NAMES,
6. \*PFW\_NETWORK\_NAMES;

**dwNumEntries:**  Specifies the number of network names in the *wszNames* array.

**wszNames:**  An array of pointers to null-terminated [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) strings representing the network DNS suffix as specified in the network interface DNS suffix. Each pointer string MUST NOT be NULL , the string MUST NOT contain the pipe (|) character, MUST be a string at least 1 character long, and MUST NOT be greater than or equal to 255 characters.

### FW\_RULE2\_20

This structure represents a firewall rule that is used by the 2.20 binary version servers and clients (see sections [1.7](#Section_6b14b8f991ab4f09873f9e2334196cdc) and [2.2.41](#Section_faf4ffbe1d5140adae902230f2c0b6a9)). The fields of this structure are identical to the FW\_RULE structure and their meanings are covered in section [2.2.36](#Section_8c008258166d46d49090f2ffaa01be4b).

1. typedef struct \_tag\_FW\_RULE2\_20 {
2. struct \_tag\_FW\_RULE\* pNext;
3. unsigned short wSchemaVersion;
4. [string, range(1, 512), ref] wchar\_t\* wszRuleId;
5. [string, range(1, 10001)] wchar\_t\* wszName;
6. [string, range(1, 10001)] wchar\_t\* wszDescription;
7. unsigned long dwProfiles;
8. [range(FW\_DIR\_INVALID, FW\_DIR\_OUT)]
9. FW\_DIRECTION Direction;
10. [range(0, 256)] unsigned short wIpProtocol;
11. [switch\_type(unsigned short), switch\_is(wIpProtocol)]
12. union {
13. [case(6,17)]
14. struct {
15. FW\_PORTS LocalPorts;
16. FW\_PORTS RemotePorts;
17. };
18. [case(1)]
19. FW\_ICMP\_TYPE\_CODE\_LIST V4TypeCodeList;
20. [case(58)]
21. FW\_ICMP\_TYPE\_CODE\_LIST V6TypeCodeList;
22. [default] ;
23. };
24. FW\_ADDRESSES LocalAddresses;
25. FW\_ADDRESSES RemoteAddresses;
26. FW\_INTERFACE\_LUIDS LocalInterfaceIds;
27. unsigned long dwLocalInterfaceTypes;
28. [string, range(1, 10001)] wchar\_t\* wszLocalApplication;
29. [string, range(1, 10001)] wchar\_t\* wszLocalService;
30. [range(FW\_RULE\_ACTION\_INVALID, FW\_RULE\_ACTION\_MAX)]
31. FW\_RULE\_ACTION Action;
32. unsigned short wFlags;
33. [string, range(1, 10001)] wchar\_t\* wszRemoteMachineAuthorizationList;
34. [string, range(1, 10001)] wchar\_t\* wszRemoteUserAuthorizationList;
35. [string, range(1, 10001)] wchar\_t\* wszEmbeddedContext;
36. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
37. FW\_RULE\_STATUS Status;
38. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX)]
39. FW\_RULE\_ORIGIN\_TYPE Origin;
40. [string, range(1, 10001)] wchar\_t\* wszGPOName;
41. unsigned long Reserved;
42. [size\_is((Reserved & FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA) ? 1 : 0)]
43. PFW\_OBJECT\_METADATA pMetaData;
44. [string, range(1, 10001)] WCHAR\* wszLocalUserAuthorizationList;
45. [string, range(1, 10001)] WCHAR\* wszPackageId;
46. [string, range(1, 10001)] WCHAR\* wszLocalUserOwner;
47. unsigned long dwTrustTupleKeywords;
48. } FW\_RULE2\_20,
49. \*PFW\_RULE2\_20;

### FW\_RULE\_FLAGS2

This enumeration represents flags that can be specified in firewall rules of section [2.2.36](#Section_8c008258166d46d49090f2ffaa01be4b).

1. typedef enum \_tag\_FW\_RULE\_FLAGS2
2. {
3. FW\_RULE\_FLAGS2\_NONE = 0x0000,
4. FW\_RULE\_FLAGS2\_SYSTEMOS\_ONLY = 0x0001,
5. FW\_RULE\_FLAGS2\_GAMEOS\_ONLY = 0x0002,
6. FW\_RULE\_FLAGS2\_DEVMODE = 0x0004,
7. FW\_RULE\_FLAGS\_MAX\_V2\_26 = 0x0008,
8. FW\_RULE\_FLAGS2\_NOT\_USED\_VALUE\_8 = 0x0008,
9. FW\_RULE\_FLAGS2\_EMPTY\_REMOTENAME = 0x0010,
10. FW\_RULE\_FLAGS2\_NOT\_REMOTENAME = 0x0020,
11. FW\_RULE\_FLAGS2\_NOT\_USED\_VALUE\_64 = 0x0040,
12. FW\_RULE\_FLAGS2\_CALLOUT\_AND\_AUDIT = 0x0080,
13. FW\_RULE\_FLAGS2\_MAX = 0x0100
14. }FW\_RULE\_FLAGS2;

**FW\_RULE\_FLAGS2\_NONE**: This value means that none of the following flags are set. It is defined for simplicity in writing IDL definitions and code.

**FW\_RULE\_FLAGS2\_SYSTEMOS\_ONLY**: This value is not used over the wire.

**FW\_RULE\_FLAGS2\_GAMEOS\_ONLY**: This value is not used over the wire.

**FW\_RULE\_FLAGS2\_DEVMODE**: This value is not used over the wire.

**FW\_RULE\_FLAGS\_MAX\_V2\_26**: This value and values that exceed this value are not valid and MUST NOT be used by servers and clients with schema version 0x021A and earlier. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 0x0008.

**FW\_RULE\_FLAGS2\_NOT\_USED\_VALUE\_8**: This value is not used over the wire.

**FW\_RULE\_FLAGS2\_NOT\_USED\_VALUE\_64**: This value is not used.

**FW\_RULE\_FLAGS2\_CALLOUT\_AND\_AUDIT**: Rules that specify this value will perform the indicated action and will then record an audit event to validate that such an action occurred.[<22>](#Appendix_A_22" \o "Product behavior note 22)

**FW\_RULE\_FLAGS2\_MAX**: This value and values that exceed this value are not valid and MUST NOT be used. It is defined for simplicity in writing IDL definitions and code. This symbolic constant has a value of 0x0100.

### FW\_RULE2\_24

This structure represents a firewall rule that is used by the 2.24 binary version servers and clients (see sections [1.7](#Section_6b14b8f991ab4f09873f9e2334196cdc) and [2.2.41](#Section_faf4ffbe1d5140adae902230f2c0b6a9)). Except as noted below, the fields of this structure are identical to the FW\_RULE structure and their meanings are covered in section [2.2.36](#Section_8c008258166d46d49090f2ffaa01be4b).

1. typedef struct \_tag\_FW\_RULE2\_24 {
2. struct \_tag\_FW\_RULE2\_24\* pNext;
3. unsigned short wSchemaVersion;
4. [string, range(1, 512), ref] wchar\_t\* wszRuleId;
5. [string, range(1, 10001)] wchar\_t\* wszName;
6. [string, range(1, 10001)] wchar\_t\* wszDescription;
7. unsigned long dwProfiles;
8. [range(FW\_DIR\_INVALID, FW\_DIR\_OUT)]
9. FW\_DIRECTION Direction;
10. [range(0, 256)] unsigned short wIpProtocol;
11. [switch\_type(unsigned short), switch\_is(wIpProtocol)]
12. union {
13. [case(6,17)]
14. struct {
15. FW\_PORTS LocalPorts;
16. FW\_PORTS RemotePorts;
17. };
18. [case(1)]
19. FW\_ICMP\_TYPE\_CODE\_LIST V4TypeCodeList;
20. [case(58)]
21. FW\_ICMP\_TYPE\_CODE\_LIST V6TypeCodeList;
22. [default]  ;
23. };
24. FW\_ADDRESSES LocalAddresses;
25. FW\_ADDRESSES RemoteAddresses;
26. FW\_INTERFACE\_LUIDS LocalInterfaceIds;
27. unsigned long dwLocalInterfaceTypes;
28. [string, range(1, 10001)] wchar\_t\* wszLocalApplication;
29. [string, range(1, 10001)] wchar\_t\* wszLocalService;
30. [range(FW\_RULE\_ACTION\_INVALID, FW\_RULE\_ACTION\_MAX)]
31. FW\_RULE\_ACTION Action;
32. unsigned short wFlags;
33. [string, range(1, 10001)] wchar\_t\* wszRemoteMachineAuthorizationList;
34. [string, range(1, 10001)] wchar\_t\* wszRemoteUserAuthorizationList;
35. [string, range(1, 10001)] wchar\_t\* wszEmbeddedContext;
36. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
37. FW\_RULE\_STATUS Status;
38. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX)]
39. FW\_RULE\_ORIGIN\_TYPE Origin;
40. [string, range(1, 10001)] wchar\_t\* wszGPOName;
41. unsigned long Reserved;
42. [size\_is((Reserved & FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA) ? 1 : 0)]
43. PFW\_OBJECT\_METADATA pMetaData;
44. [string, range(1, 10001)] WCHAR\* wszLocalUserAuthorizationList;
45. [string, range(1, 10001)] WCHAR\* wszPackageId;
46. [string, range(1, 10001)] WCHAR\* wszLocalUserOwner;
47. unsigned long dwTrustTupleKeywords;
48. FW\_NETWORK\_NAMES OnNetworkNames;
49. [string, range(1, 10001)] WCHAR\* wszSecurityRealmId;
50. } FW\_RULE2\_24,
51. \*PFW\_RULE2\_24;

### FW\_RULE2\_25

This structure represents a firewall rule that is used by the 2.25 binary version servers and clients (see sections [1.7](#Section_6b14b8f991ab4f09873f9e2334196cdc) and [2.2.41](#Section_faf4ffbe1d5140adae902230f2c0b6a9)). The fields of this structure are identical to the FW\_RULE structure and their meanings are covered in section [2.2.36](#Section_8c008258166d46d49090f2ffaa01be4b).

1. typedef struct \_tag\_FW\_RULE2\_25 {
2. struct \_tag\_FW\_RULE\* pNext;
3. unsigned short wSchemaVersion;
4. [string, range(1, 10001), ref] wchar\_t\* wszRuleId;
5. [string, range(1, 10001)] wchar\_t\* wszName;
6. [string, range(1, 10001)] wchar\_t\* wszDescription;
7. unsigned long dwProfiles;
8. [range(FW\_DIR\_INVALID, FW\_DIR\_OUT)]
9. FW\_DIRECTION Direction;
10. [range(0, 256)] unsigned short wIpProtocol;
11. [switch\_type(unsigned short), switch\_is(wIpProtocol)]
12. union {
13. [case(6,17)]
14. struct {
15. FW\_PORTS LocalPorts;
16. FW\_PORTS RemotePorts;
17. };
18. [case(1)]
19. FW\_ICMP\_TYPE\_CODE\_LIST V4TypeCodeList;
20. [case(58)]
21. FW\_ICMP\_TYPE\_CODE\_LIST V6TypeCodeList;
22. [default]  ;
23. };
24. FW\_ADDRESSES LocalAddresses;
25. FW\_ADDRESSES RemoteAddresses;
26. FW\_INTERFACE\_LUIDS LocalInterfaceIds;
27. unsigned long dwLocalInterfaceTypes;
28. [string, range(1, 10001)] wchar\_t\* wszLocalApplication;
29. [string, range(1, 10001)] wchar\_t\* wszLocalService;
30. [range(FW\_RULE\_ACTION\_INVALID, FW\_RULE\_ACTION\_MAX)]
31. FW\_RULE\_ACTION Action;
32. unsigned short wFlags;
33. [string, range(1, 10001)] wchar\_t\* wszRemoteMachineAuthorizationList;
34. [string, range(1, 10001)] wchar\_t\* wszRemoteUserAuthorizationList;
35. [string, range(1, 10001)] wchar\_t\* wszEmbeddedContext;
36. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
37. FW\_RULE\_STATUS Status;
38. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX)]
39. FW\_RULE\_ORIGIN\_TYPE Origin;
40. [string, range(1, 10001)] wchar\_t\* wszGPOName;
41. unsigned long Reserved;
42. [size\_is((Reserved & FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA) ? 1 : 0)]
43. PFW\_OBJECT\_METADATA pMetaData;
44. [string, range(1, 10001)] WCHAR\* wszLocalUserAuthorizationList;
45. [string, range(1, 10001)] WCHAR\* wszPackageId;
46. [string, range(1, 10001)] WCHAR\* wszLocalUserOwner;
47. Unsigned long dwTrustTupleKeywords;
48. FW\_NETWORK\_NAMES OnNetworkNames;
49. [string, range(1, 10001)] WCHAR\* wszSecurityRealmId;
50. unsigned short wFlags2;
51. } FW\_RULE2\_25,
52. \*PFW\_RULE2\_25;

### FW\_RULE2\_26

This structure represents a firewall rule that is used by 2.26 binary policy version servers and clients (section [1.7](#Section_6b14b8f991ab4f09873f9e2334196cdc) and section [2.2.41](#Section_faf4ffbe1d5140adae902230f2c0b6a9)). Definitions for the FW\_RULE2\_26 structure fields are described in section [2.2.36](#Section_8c008258166d46d49090f2ffaa01be4b).

1. typedef struct \_tag\_FW\_RULE2\_26 {
2. struct \_tag\_FW\_RULE\* pNext;
3. unsigned short wSchemaVersion;
4. [string, range(1, 10001), ref] wchar\_t\* wszRuleId;
5. [string, range(1, 10001)] wchar\_t\* wszName;
6. [string, range(1, 10001)] wchar\_t\* wszDescription;
7. unsigned long dwProfiles;
8. [range(FW\_DIR\_INVALID, FW\_DIR\_OUT)]
9. FW\_DIRECTION Direction;
10. [range(0, 256)] unsigned short wIpProtocol;
11. [switch\_type(unsigned short), switch\_is(wIpProtocol)]
12. union {
13. [case(6,17)]
14. struct {
15. FW\_PORTS LocalPorts;
16. FW\_PORTS RemotePorts;
17. };
18. [case(1)]
19. FW\_ICMP\_TYPE\_CODE\_LIST V4TypeCodeList;
20. [case(58)]
21. FW\_ICMP\_TYPE\_CODE\_LIST V6TypeCodeList;
22. [default]  ;
23. };
24. FW\_ADDRESSES LocalAddresses;
25. FW\_ADDRESSES RemoteAddresses;
26. FW\_INTERFACE\_LUIDS LocalInterfaceIds;
27. unsigned long dwLocalInterfaceTypes;
28. [string, range(1, 10001)] wchar\_t\* wszLocalApplication;
29. [string, range(1, 10001)] wchar\_t\* wszLocalService;
30. [range(FW\_RULE\_ACTION\_INVALID, FW\_RULE\_ACTION\_MAX)]
31. FW\_RULE\_ACTION Action;
32. unsigned short wFlags;
33. [string, range(1, 10001)] wchar\_t\* wszRemoteMachineAuthorizationList;
34. [string, range(1, 10001)] wchar\_t\* wszRemoteUserAuthorizationList;
35. [string, range(1, 10001)] wchar\_t\* wszEmbeddedContext;
36. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
37. FW\_RULE\_STATUS Status;
38. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX)]
39. FW\_RULE\_ORIGIN\_TYPE Origin;
40. [string, range(1, 10001)] wchar\_t\* wszGPOName;
41. unsigned long Reserved;
42. [size\_is((Reserved & FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA) ? 1 : 0)]
43. PFW\_OBJECT\_METADATA pMetaData;
44. [string, range(1, 10001)] WCHAR\* wszLocalUserAuthorizationList;
45. [string, range(1, 10001)] WCHAR\* wszPackageId;
46. [string, range(1, 10001)] WCHAR\* wszLocalUserOwner;
47. Unsigned long dwTrustTupleKeywords;
48. FW\_NETWORK\_NAMES OnNetworkNames;
49. [string, range(1, 10001)] WCHAR\* wszSecurityRealmId;
50. unsigned short wFlags2;
51. FW\_NETWORK\_NAMES RemoteOutServerNames;
52. } FW\_RULE2\_26,

\*PFW\_RULE2\_26;

# Protocol Details

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

## Server Details

### 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 what is described in this document.

**GlobalConfiguration**: A table of policy configuration options where each entry contains:

* **GlobalOptionType**: This identifies the global option type. The global option types supported by this protocol are defined by the data type [FW\_GLOBAL\_CONFIG (section 2.2.41)](#Section_faf4ffbe1d5140adae902230f2c0b6a9).
* **GlobalOptionValue**: This contains the current value for this global option type. See FW\_GLOBAL\_CONFIG (section 2.2.41) for details about the data type used to represent each global option type.

**ProfileConfiguration**: A table of policy configuration options that apply to a single profile where each entry contains:

* **ProfileOptionType**: This identifies the profile option type. The profile option types supported by this protocol are defined by the data type [FW\_PROFILE\_CONFIG (section 2.2.37)](#Section_5a6e0d39802d456bb483c7360566fcdd).
* **ProfileOptionValue**: This contains the current value for this profile option type. See FW\_PROFILE\_CONFIG (section 2.2.37) for details about the data type used to represent each profile option type.

**ProfileConfigurationTable**: This is a table of the **ProfileConfiguration** objects for each profile type, where each entry contains:

* **ProfileType**: This identifies the profile to which the configuration applies. The profile types supported by this protocol are defined by the data type [FW\_PROFILE\_TYPE (section 2.2.2)](#Section_7704e238174d4a5eb8095f3787dd8acc). This table only contains entries for the domain, private, and public profiles.
* **ProfileConfiguration**: This contains the configuration options for that profile.

**FirewallRule**: This describes a firewall rule, which is defined in this protocol by the data type [FW\_RULE (section 2.2.36)](#Section_8c008258166d46d49090f2ffaa01be4b).

**FirewallRules**: A set of **FirewallRule** objects.

**AuthenticationSet**: This describes an authentication set, which is defined in this protocol by the data type [FW\_AUTH\_SET (section 2.2.64)](#Section_bbec7ab70a1d49a49c8ea4784bbe6f16). This object contains two additional properties:

* **IsAuthPrimary**: A Boolean value indicating that this is a primary set. The Phase 1 and Phase 2 primary authentication sets are identified by well-known set IDs as specified in section 2.2.64. Note that the value of this property can always be derived from the set ID; it is described separately solely for convenience.

Primary authentication sets differ from other authentication sets in that they are guaranteed to exist in the **GroupPolicyRSoPStore** and the **LocalStore**. If the administrator does not explicitly add the primary sets, the server initializes them to default values. See section [3.1.3](#Section_e8924ac5aa4a41d1bf654f46b3d399aa) for details.

Although this protocol imposes no limitations on how administrators use the primary authentication sets, the intent is to decouple management of authentication settings from management of connection security and main mode rules. In this model, most rules do not use unique authentication sets, but instead reference the primary sets.

* **IsAuthConfigured**: A Boolean value indicating that this set was configured by an administrator rather than initialized to hard-coded values. This property MUST be ignored if **IsAuthPrimary** is false.

**AuthenticationSets**: A set of **AuthenticationSet** objects.

**CryptoSet**: This describes a crypto set, which is defined in this protocol by the data type [FW\_CRYPTO\_SET (section 2.2.73)](#Section_a468fe9e113b4155a63d0db3aac12619). This object contains two additional properties:

* **IsCryptoPrimary**: A Boolean value indicating that this is a primary set. The Phase 1 and Phase 2 primary crypto sets are identified by well-known set IDs as specified in section 2.2.73. Note that the value of this property can always be derived from the set ID; it is described separately solely for convenience.

Primary crypto sets differ from other crypto sets in that they are guaranteed to exist in the **GroupPolicyRSoPStore** and the **LocalStore**. If the administrator does not explicitly add the primary sets, the server initializes them to default values. See section 3.1.3 for details.

Although this protocol imposes no limitations on how administrators use the primary crypto sets, the intent is to decouple management of crypto settings from management of connection security rules. In this model, most rules do not use unique crypto sets, but instead reference the primary sets.

* **IsCryptoConfigured**: A Boolean value indicating this set was configured by an administrator rather than initialized to default values by the server. This property MUST be ignored if **IsCryptoPrimary** is false.

**CryptoSets**: A set of **CryptoSet** objects.

**ConnectionSecurityRule**: This describes a connection security rule, which is defined in this protocol by the data type [FW\_CS\_RULE (section 2.2.54)](#Section_0d0641105f2e4b68aa63032c6cd5e4c6). A **ConnectionSecurityRule** contains references to **AuthenticationSet** and **CryptoSet** objects in the store.

**ConnectionSecurityRules**: A set of **ConnectionSecurityRule** objects.

**MainModeRule**: A main mode rule, which is defined in this protocol by the data type [FW\_MM\_RULE (section 2.2.84)](#Section_4b3fc163fede434c90cb557992caa5da). A **MainModeRule** contains a reference to an **AuthenticationSet** in the store.

**MainModeRules**: A set of **MainModeRule** objects.

**PolicyStore**: This represents a collection of policy settings. A **PolicyStore** contains a single instance of each of the following objects:

* **GlobalConfiguration**
* **ProfileConfigurationTable**
* **FirewallRules**
* **AuthenticationSets**
* **CryptoSets**
* **ConnectionSecurityRules**
* **MainModeRules**

**PolicyStoreConnection**: This represents a client connection to a **PolicyStore**. It maintains the association between the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) connection and the **PolicyStore** being managed. It contains the following fields:

* **StoreType**: The type of store being managed, which is defined in this protocol by the data type [FW\_STORE\_TYPE (section 2.2.1)](#Section_37ebed958abf472c8b4b7a510a2a6baa). This value MUST be FW\_STORE\_TYPE\_GP\_RSOP, FW\_STORE\_TYPE\_LOCAL, FW\_STORE\_TYPE\_DYNAMIC, or FW\_STORE\_TYPE\_DEFAULTS.
* **BinaryVersion**: An unsigned integer representing the binary version of the RPC interface used by the client. This value MUST be a valid Protocol Version (see section [1.7](#Section_6b14b8f991ab4f09873f9e2334196cdc)).

**PortInUse**: This represents an Internet Protocol transport layer port that is currently in use by an [**endpoint**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee) on the local computer. It contains the following fields:

* **AddressFamily**: The address family of the endpoint. This MUST be IPv4 or IPv6.
* **TransportProtocol**: The transport protocol used by the endpoint. This MUST be [**TCP**](#gt_b08d36f6-b5c6-4ce4-8d2d-6f2ab75ea4cb) or UDP.
* **PortNumber**: The port number used by the transport protocol. This MUST be an integer in the range of 1 to 65535 inclusive.
* **IsDynamicRPC**: A Boolean value indicating that the port is in use by an RPC server, and that the port was randomly selected at runtime.
* **IsRPCEndpointMapper**: A Boolean value indicating that the port is in use by the RPC endpoint mapper.
* **IsTeredo**: A Boolean value indicating that the port is in use by Teredo.
* **IsIPTLSIn**: A Boolean value indicating that the port is in use for inbound IP-TLS connections.
* **IsIPTLSOut**: A Boolean value indicating that the port is in use for outbound IP-TLS connections.
* **NATTraversalRequested**: A Boolean value indicating that the application that created the endpoint is designed to take advantage of IPv6 NAT traversal capabilities (Teredo, for example).

**PortsInUse**: A set of **PortInUse** objects. The contents of the **PortsInUse** collection are determined solely through the [AddPortInUse (section 3.1.6.1)](#Section_ad9fbd21d8c344cea24941608e6c4a39) and [DeletePortInUse (section 3.1.6.2)](#Section_bb63abaa14e34a24a5552a4142a25bb5) abstract interfaces.

**TrustTuple**: This describes Internet Protocol transport layer traffic that is currently being sent or received by an endpoint on the local computer. It contains the following fields:

* **AddressFamily**: The address family of the traffic. This MUST be IPv4 or IPv6.
* **TransportProtocol**: The transport protocol used by the traffic. This MUST be TCP or UDP.
* **LocalAddress**: The local IPv4 or IPv6 address of the traffic.
* **RemoteAddress**: The remote IPv4 or IPv6 address of the traffic.
* **LocalPortNumber**: The local port number used by the transport protocol. This MUST be an integer in the range of 1 to 65535 inclusive.
* **RemotePortNumber**: The remote port number used by the transport protocol. This MUST be an integer in the range of 1 to 65535 inclusive.
* **IsProximity**: A Boolean value indicating that the remote endpoint is located in close physical proximity to the local computer.
* **IsProximitySharing**: A Boolean value indicating that the traffic is used to share data with a remote endpoint located in close physical proximity to the local computer.
* **IsWFDPrint**: A Boolean value indicating that the traffic is used to send data to a printer over Wi-Fi Direct.
* **IsWFDDisplay**: A Boolean value indicating that the traffic is used to mirror or extend the local computer screen with a display device over Wi-Fi Direct.
* **IsWFDDevices**: A Boolean value indicating that the traffic is used to send data to a device over Wi-Fi Direct.
* **IsWFDMaUsbWirelessDocking**: A Boolean value indicating that the traffic is used to send data in Media Agnostic USB for Wireless Docking scenarios.

**TrustTuples**: A set of **TrustTuple** objects. The contents of the **TrustTuples** collection are determined solely through the AddTrustTuple (section [3.1.6.7](#Section_4d0dfae28ff7491d8a08a44cf0503440)) and DeleteTrustTuple (section [3.1.6.8](#Section_3f2b41978ce04a53bfb61d977a94cc5e)) abstract interfaces.

**MSFASPServer**: This represents the state maintained by a server that implements this protocol. It contains multiple instances of **PolicyStore**. These instances are identified by the data type FW\_STORE\_TYPE (section 2.2.1). The server maintains the following objects:

* **GroupPolicyRSoPStore**: An instance of **PolicyStore** corresponding to FW\_STORE\_TYPE\_GP\_RSOP. The state of this object MUST be maintained in persistent storage.
* **LocalStore**: An instance of **PolicyStore** corresponding to FW\_STORE\_TYPE\_LOCAL. The state of this object MUST be maintained in persistent storage.
* **DynamicStore**: An instance of **PolicyStore** corresponding to FW\_STORE\_TYPE\_DYNAMIC.
* **DefaultsStore**: An instance of **PolicyStore** corresponding to FW\_STORE\_TYPE\_DEFAULTS. The state of this object MUST be maintained in persistent storage. The name **DefaultsStore** was chosen to maintain consistent naming between the ADM and the data types and operations defined in this protocol. However, this element is not used to store default settings in the traditional sense. Instead, it is used to store a known good configuration for the **LocalStore**. The administrator can explicitly revert the **LocalStore** to these settings by invoking [RRPC\_FWRestoreDefaults (section 3.1.4.3)](#Section_764123cab72743229fe1de1fd859fac1). Otherwise, the contents of this store are ignored.
* **PortsInUse**: This represents the set of all **PortInUse** objects managed by the server. Elements are added and deleted from this set through the abstract interfaces AddPortInUse and DeletePortInUse.
* **TrustTuples**: This represents the set of all **TrustTuple** objects managed by the server. Elements are added and deleted from this set through the abstract interfaces AddTrustTuple and DeleteTrustTuple.

### Timers

No protocol timer events are required on the server side other than the timers required by the underlying [**RPC transport**](#gt_c2eeb200-3cd0-4916-966e-d7d6bff1737a), as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15).

### Initialization

The server initializes when the server host machine starts. The server MUST restore the state of the **GroupPolicyRSoPStore**, the **LocalStore**, and the **DefaultsStore** from persistent storage. The order in which the stores are loaded does not matter. The **PortsInUse** collection and the **TrustTuples** collection MUST be initialized to an empty set.

The server MUST ensure that **LocalStore** and **GroupPolicyRSoPStore** contain the Phase 1 and Phase 2 primary **AuthenticationSet** objects. If either of the primary sets is missing, the server MUST create a new instance and set the corresponding **IsAuthConfigured** property to false. The values used to initialize the new instances are implementation-specific.[<23>](#Appendix_A_23" \o "Product behavior note 23)

The server MUST ensure that **LocalStore** and **GroupPolicyRSoPStore** contain the Phase 1 and Phase 2 primary **CryptoSet** objects. If either of the primary sets is missing, the server MUST create a new instance and set the corresponding **IsCryptoConfigured** property to false. The values used to initialize the new instances are implementation-specific.[<24>](#Appendix_A_24" \o "Product behavior note 24)

The server MUST merge **GroupPolicyRSoPStore** and **LocalStore** and use the result to initialize **DynamicStore**. The merge logic is as follows:

* For the **GlobalConfiguration** and **ProfileConfiguration** options, if an option is configured in only one store, that value MUST be used. If an option is configured in neither store, the option MUST be initialized to an implementation-specific[<25>](#Appendix_A_25" \o "Product behavior note 25) default value. If an option is configured in both stores, the values MUST be merged according to the merge law for that option. The merge laws for **GlobalConfiguration** and **ProfileConfiguration** options are specified in sections [2.2.41](#Section_faf4ffbe1d5140adae902230f2c0b6a9) and [2.2.37](#Section_5a6e0d39802d456bb483c7360566fcdd) respectively.
* For **FirewallRules**, **ConnectionSecurityRules**, and **MainModeRules**, all the rules from both stores MUST be combined and added to **DynamicStore**.
* For **AuthenticationSets**, if a primary set in **GroupPolicyRSoPStore** has **IsAuthConfigured** set to true, that set MUST be added to **DynamicStore** and the corresponding set in **LocalStore** MUST be ignored. Otherwise, the primary set from **LocalStore** MUST be used. For all other sets (that is, the sets where **IsAuthPrimary** is false), the sets from both stores MUST be combined and added to **DynamicStore**.
* For **CryptoSets**, if a primary set in **GroupPolicyRSoPStore** has **IsCryptoConfigured** set to true, that set MUST be added to **DynamicStore** and the corresponding set in **LocalStore** MUST be ignored. Otherwise, the primary set from **LocalStore** MUST be used. For all other sets (that is, the sets where **IsCryptoPrimary** is false), the sets from both stores MUST be combined and added to **DynamicStore**.

After the merge is complete, the server MUST invoke the abstract interface [SetEffectiveFirewallPolicy (section 3.1.6.6)](#Section_0b159c9216fa42a69a33b68f0e040a98) with the contents of **DynamicStore**. It MUST register the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) interface and begin listening on the RPC [**endpoint**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee) as specified in section [2.1](#Section_81eb95d6df6349b6905265c99664e71f).

### Message Processing Events and Sequencing Rules

This protocol MUST indicate to the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) runtime that it is to perform a strict [**Network Data Representation (NDR)**](#gt_9ebf9540-2c31-43bc-bc56-4a932faabf2d) data consistency check at target level 6.0, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15).

This protocol MUST indicate to the RPC runtime, via the strict\_context\_handle attribute, that it is to reject the use of context handles that are created by using a different method of RPC interface than this one, as specified in [MS-RPCE] section 3.

Because the server makes access control decisions as part of message processing, the client MUST authenticate to the server as specified in section [2.1](#Section_81eb95d6df6349b6905265c99664e71f). The server MUST verify that the client is authorized to perform the requested operation. The server MUST retrieve the client's identity token by invoking the abstract interface GetRpcImpersonationAccessToken() as specified in [MS-RPCE] section 3.3.3.4.3.1. The server implementation maintains a list of authorized clients. The protocol has no methods for reading or setting that list. If the client invoking the method is not on the authorized list, the server MUST fail the call and return an error code of ERROR\_ACCESS\_DENIED (5).[<26>](#Appendix_A_26" \o "Product behavior note 26)

**Methods in RPC Opnum Order**

| Method | Description |
| --- | --- |
| [RRPC\_FWOpenPolicyStore](#Section_2157e39a1bf04e0cabae0811fd918b11) | This method requests the server to open a specified policy store.  Opnum: 0 |
| [RRPC\_FWClosePolicyStore](#Section_203dc96f2e74449aae18aaae0689c092) | This method receives an opened store handle and closes it, freeing any resources that were allocated by the server-to-server operations on the opened store.  Opnum: 1 |
| [RRPC\_FWRestoreDefaults](#Section_764123cab72743229fe1de1fd859fac1) | This method erases the local policy store and replaces it with the default policy that the server host had out of the box after installation. After the method returns, the local store contains exactly the same policy as it did after installation.  Opnum: 2 |
| [RRPC\_FWGetGlobalConfig](#Section_10c07a562a4f4f068510c6f3bfe5ca4a) | This method retrieves the value of a global policy configuration option. The client specifies to the server from what store this value MUST be retrieved and in what specific configuration option it is interested.  Opnum: 3 |
| [RRPC\_FWSetGlobalConfig](#Section_45676b5c298b413b8fac0770c3def938) | This method modifies the value of a global policy configuration option. The client specifies to the server in what store this value MUST be written and what specific configuration option it is interested in modifying.  Opnum: 4 |
| [RRPC\_FWAddFirewallRule](#Section_4d7baada78fd469a9c292885e5d3b5d0) | This method requests the server to add the specified firewall rule in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 5 |
| [RRPC\_FWSetFirewallRule](#Section_676f90d026b44ed1a5eb1f88db8676b4) | This method requests the server to modify the specified firewall rule in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 6 |
| [RRPC\_FWDeleteFirewallRule](#Section_e1ad0b524218420bb982d5035f6c8da3) | This method requests the server to delete the specified firewall rule in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 7 |
| [RRPC\_FWDeleteAllFirewallRules](#Section_ca2da2f2b8ef4981abbafed5c713990d) | This method deletes all firewall rules in the firewall linked list of the memory representation of the store being modified.  Opnum: 8 |
| [RRPC\_FWEnumFirewallRules](#Section_469f8466bb5c4af097a8335f648c70d7) | This method requests the server to return all the firewall rules contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all the firewall rule objects.  Opnum: 9 |
| [RRPC\_FWGetConfig](#Section_debe35e3cba149ceb331374d4e88d0ae) | This method retrieves the value of a profile configuration option. The client specifies to the server from what store and profile this value MUST be retrieved and in what specific configuration option it is interested.  Opnum: 10 |
| [RRPC\_FWSetConfig](#Section_8621c5249ccc495c9c80a0830282ccf8) | This method modifies the value of a profile configuration option. The client specifies to the server in what store and profile this value MUST be written and what specific configuration option it is interested in modifying.  Opnum: 11 |
| [RRPC\_FWAddConnectionSecurityRule](#Section_4e31fcb5b70148b58a45d70bf85281ab) | This method requests the server to add the connection security rule in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 12 |
| [RRPC\_FWSetConnectionSecurityRule](#Section_c66ad9a72ee44d82a62e73ce5599c05c) | This method requests the server to modify the specified connection security rule in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 13 |
| [RRPC\_FWDeleteConnectionSecurityRule](#Section_b470f66f926247479be719f8aa92a611) | This method requests the server to delete the specified connection security rule in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 14 |
| [RRPC\_FWDeleteAllConnectionSecurityRules](#Section_ee4f85d9c07449658508c8ec96ae3b77) | This method requests the server to delete all the connection security rules in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 15 |
| [RRPC\_FWEnumConnectionSecurityRules](#Section_f0f85db355104b73a4cb76bd440eaa06) | This method requests the server to return all the connection security rules contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all the connection security rule objects.  Opnum: 16 |
| [RRPC\_FWAddAuthenticationSet](#Section_c1dbe17a32cf4910843b7d2690e9087a) | This method requests the server to add the authentication set in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 17 |
| [RRPC\_FWSetAuthenticationSet](#Section_5a5b0f3ad8ac45608a6a485ece2cbb6c) | This method requests the server to modify the specified authentication set in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 18 |
| [RRPC\_FWDeleteAuthenticationSet](#Section_b143f0f85f274595b5310197fc491d0b) | This method requests the server to delete the specified authentication set in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 19 |
| [RRPC\_FWDeleteAllAuthenticationSets](#Section_7d8f82f7d9284ebc8afc576751e2e8d2) | This method requests the server to delete all the authentication sets of a specific [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) phase in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 20 |
| [RRPC\_FWEnumAuthenticationSets](#Section_2dc9c31538e842d69c88927dea828314) | This method requests the server to return all the authentication sets of the specified IPsec phase contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of these objects.  Opnum: 21 |
| [RRPC\_FWAddCryptoSet](#Section_0399561877894a00828cd8c497b643e3) | This method adds a cryptographic set in the cryptographic linked list of the memory representation of the store being modified.  Opnum: 22 |
| [RRPC\_FWSetCryptoSet](#Section_423d057609384c818c9beacafc69504c) | This method requests the server to modify the specified cryptographic set in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 23 |
| [RRPC\_FWDeleteCryptoSet](#Section_6c5b4c64545c4b108208414de327ead9) | This method requests the server to delete the specified cryptographic set in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 24 |
| [RRPC\_FWDeleteAllCryptoSets](#Section_67cf747ee0054630932928887df1e09d) | This method requests the server to delete all the cryptographic sets of a specific IPsec phase in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 25 |
| [RRPC\_FWEnumCryptoSets](#Section_63e911eafbca409689a95d84c37daa56) | This method requests the server to return all the cryptographic sets of the specified IPsec phase contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all these cryptographic objects.  Opnum: 26 |
| [RRPC\_FWEnumPhase1SAs](#Section_c97e073507544e688c82fb3b4275d085) | This method requests the server to return all the [**security associations**](#gt_67cbf867-7a49-41f3-a68f-37b5f9035acb) of the IPsec first-negotiation phase contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all these security associations.  Opnum: 27 |
| [RRPC\_FWEnumPhase2SAs](#Section_2c4f14fc0fee41f0b1f3e0082c929995) | This method requests the server to return all the security associations of the IPsec second-negotiation phase contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all these security associations.  Opnum: 28 |
| [RRPC\_FWDeletePhase1SAs](#Section_0666c872b08f4b82a263e1935b763141) | This method requests the server to delete all the IPsec first negotiation phase security associations that match the specified [**endpoints**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee).  Opnum: 29 |
| [RRPC\_FWDeletePhase2SAs](#Section_1b6a54e35aba4bc49b9629b5514d585f) | This method requests the server to delete all the IPsec second negotiation phase security associations that match the specified endpoints.  Opnum: 30 |
| [RRPC\_FWEnumProducts](#Section_b7a0c4f8f7a948f1b542a2937e269ca3) | This method requests the server to return all the registered third-party software components registered with the firewall and advanced security component.  Opnum: 31 |
| [RRPC\_FWAddMainModeRule](#Section_40973e9f20c84e7b8f96f2664a622f03) | This method requests the server to add the main mode rule in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 32 |
| [RRPC\_FWSetMainModeRule](#Section_268116a5af2b407b850c1380bece5169) | This method requests the server to modify the specified main mode rule in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 33 |
| [RRPC\_FWDeleteMainModeRule](#Section_7523b410086443278bb002d00ffa6d2b) | This method requests the server to delete the specified main mode rule in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 34 |
| [RRPC\_FWDeleteAllMainModeRules](#Section_d47dfd57c6b9448eb5a0127d3f87cd38) | This method requests the server to delete all the main mode rules in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 35 |
| [RRPC\_FWEnumMainModeRules](#Section_70ad911aa488474289549b9d60af3075) | This method requests the server to return all the main mode rules contained in the store that is referenced by the hPolicyStore handle. The method returns a linked list of all the main mode rule objects.  Opnum: 36 |
| [RRPC\_FWQueryFirewallRules](#Section_3422b6ac5b04456dbc4ffd97e1384c3a) | This method requests the server to return all the firewall rules that match the specified query object contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all the firewall rule objects.  Opnum: 37 |
| [RRPC\_FWQueryConnectionSecurityRules](#Section_b092c4bcf6d54f44b792599839c9db2f) | This method requests the server to return all the connection security rules that match the specified query object contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all the connection security rule objects.  Opnum: 38 |
| [RRPC\_FWQueryMainModeRules](#Section_dc7a4aa81cb34c1f8506e6ca581d4434) | This method requests the server to return all the main mode rules that match the specified query object contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all the main mode rule objects.  Opnum: 39 |
| [RRPC\_FWQueryAuthenticationSets](#Section_1b4f80300bb24a1ca3e57151d5f9616b) | This method requests the server to return all the authentication sets that match the specified query object contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all the authentication set objects.  Opnum: 40 |
| [RRPC\_FWQueryCryptoSets](#Section_3dc9e91dd0ee4e1ca30d70c89052de84) | This method requests the server to return all the crypto sets that match the specified query object contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all the crypto set objects.  Opnum: 41 |
| [RRPC\_FWEnumNetworks](#Section_b3f615b50a51424cab3c15b3fb350ee1) | This method requests the server to return all the networks to which the host that has the firewall and advanced security component is connected.  Opnum: 42 |
| [RRPC\_FWEnumAdapters](#Section_c5c35afeb29e433bb7bcca0ffa9adcd4) | This method requests the server to return all the network interfaces that are used by the host that has the firewall and advanced security component.  Opnum: 43 |
| [RRPC\_FWGetGlobalConfig2\_10](#Section_e479f36dc2254c1cbb67d46c43dc7c7e) | This method retrieves the value of a global policy configuration option. The client specifies to the server from what store this value MUST be retrieved and in what specific configuration option it is interested.  Opnum: 44 |
| [RRPC\_FWGetConfig2\_10](#Section_3de45ae3d1c741a88061c435cf72e480) | This method retrieves the value of a profile configuration option. The client specifies to the server from what store and profile this value MUST be retrieved and in what specific configuration option it is interested.  Opnum: 45 |
| [RRPC\_FWAddFirewallRule2\_10](#Section_5ab52f049e9d426b854560e2367fdffd) | This method requests the server to add the specified firewall rule in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 46 |
| [RRPC\_FWSetFirewallRule2\_10](#Section_27a5eb1dc5b94ccaa50263e75df80917) | This method requests the server to modify the specified firewall rule in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 47 |
| [RRPC\_FWEnumFirewallRules2\_10](#Section_77e5217a46d8440ab330679a0939c9ae) | This method requests the server to return all the firewall rules contained in the store that is referenced by the hPolicyStore handle. The method returns a linked list of all the firewall rule objects.  Opnum: 48 |
| [RRPC\_FWAddConnectionSecurityRule2\_10](#Section_b7a0d5cca02f46078681e8b5589d3451) | This method requests the server to add the connection security rule in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 49 |
| [RRPC\_FWSetConnectionSecurityRule2\_10](#Section_4229bd7163c946a1a97e3c061604d717) | This method requests the server to modify the specified connection security rule in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 50 |
| [RRPC\_FWEnumConnectionSecurityRules2\_10](#Section_8e2bce0a21e4412897473b249c0b0091) | This method requests the server to return all the connection security rules contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all the connection security rule objects.  Opnum: 51 |
| [RRPC\_FWAddAuthenticationSet2\_10](#Section_25bb2543bda546e5ba7afa6c7a356ba4) | This method requests the server to add the authentication set in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 52 |
| [RRPC\_FWSetAuthenticationSet2\_10](#Section_dfb2e973f6c64b57908afa094d90ff83) | This method requests the server to modify the specified authentication set in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 53 |
| [RRPC\_FWEnumAuthenticationSets2\_10](#Section_a788ee5f53964fd19d807e31d639b770) | This method requests the server to return all the authentication sets of the specified IPsec phase contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of these objects.  Opnum: 54 |
| [RRPC\_FWAddCryptoSet2\_10](#Section_604883c63e674140bec0ea68adf4eb53) | This method adds a cryptographic set in the cryptographic linked list of the memory representation of the store being modified.  Opnum: 55 |
| [RRPC\_FWSetCryptoSet2\_10](#Section_0b68e0c61c2844eb93a06406746d8814) | This method requests the server to modify the specified cryptographic set in the policy contained in the policy store that is referenced by the specified opened policy store handle.  Opnum: 56 |
| [RRPC\_FWEnumCryptoSets2\_10](#Section_d036149f27ed45d1a55df7881b50b3f1) | This method requests the server to return all the cryptographic sets of the specified IPsec phase that is contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all these cryptographic objects.  Opnum: 57 |
| [RRPC\_FWAddConnectionSecurityRule2\_20](#Section_222c230acb4c41b2ac44a770cd71d66c) | This method requests the server to add the specified connection security rule in the policy contained in the policy store that is referenced by the handle specified in the *hPolicyStore* parameter.  Opnum: 58 |
| [RRPC\_FWSetConnectionSecurityRule2\_20](#Section_b255739072f24d1da7db790853f2df36) | This method requests the server to modify the specified connection security rule in the policy contained in the policy store that is referenced by the handle specified in the *hPolicyStore* parameter.  Opnum: 59 |
| [RRPC\_FWEnumConnectionSecurityRules2\_20](#Section_be8c8ed3570e45c0b063bffeec096ede) | This method requests the server to return all the connection security rules contained in the store that is referenced by the *hPolicyStore* handle.  Opnum: 60 |
| [RRPC\_FWQueryConnectionSecurityRules2\_20](#Section_ce8fd2ce757a4ffa869d7ba9056bf68d) | This method requests the server to return all the connection security rules that match the specified query object that are contained in the store that is referenced by the *hPolicy* handle.  Opnum: 61 |
| [RRPC\_FWAddAuthenticationSet2\_20](#Section_b098dcdb7058470dba9df042dec221c0) | This method requests the server to add the authentication set in the policy contained in the policy store that is referenced by the handle specified in the *hPolicy* parameter.  Opnum: 62 |
| [RRPC\_FWSetAuthenticationSet2\_20](#Section_4023224c13d14df4995761b0a62937ed) | This method requests the server to modify the specified authentication set in the policy contained in the policy store that is referenced by the handle specified in the *hPolicy* parameter.  Opnum: 63 |
| [RRPC\_FWEnumAuthenticationSets2\_20](#Section_be14dfc72ce64197af85575d8e96873d) | This method requests the server to return all the authentication sets of the specified IPsec phase contained in the store that is referenced in the *hPolicy* handle. The method returns a linked list of these objects.  Opnum: 64 |
| [RRPC\_FWQueryAuthenticationSets2\_20](#Section_71836286fe1346a6a81ffed593a0fb77) | This method requests the server to return all the authentication sets that match the specified query object that are contained in the store that is referenced in the *hPolicy* handle.  Opnum: 65 |
| [RRPC\_FWAddFirewallRule2\_20](#Section_d637fee2b28d4cea8005303347de94f7) | This method requests the server to add the specified firewall rule in the policy contained in the policy store referenced by the handle that is specified in the *hPolicyStore* parameter.  Opnum: 66 |
| [RRPC\_FWSetFirewallRule2\_20](#Section_cb987e3d916f4902aaba4b99df2e3af3) | This method requests the server to modify the specified firewall rule in the policy contained in the policy store that is referenced by the handle specified in the *hPolicyStore* parameter.  Opnum: 67 |
| [RRPC\_FWEnumFirewallRules2\_20](#Section_647f1dac0be1415896c906b1e03c6bdb) | This method requests the server to return all the firewall rules contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all the firewall rule objects.  Opnum: 68 |
| [RRPC\_FWQueryFirewallRules2\_20](#Section_7d1c340631a3440fa6d47232dd8bb36e) | This method requests the server to return all the firewall rules matching the specified query object that are contained in the store referenced by the *hPolicy* handle.  Opnum: 69 |
| [RRPC\_FWAddFirewallRule2\_24](#Section_778f196ee6f347bfa3a6f6f5f7660b21) | This method requests the server to add the specified firewall rule in the policy contained in the policy store that is referenced by the handle specified in the *hPolicyStore* parameter.  Opnum: 70 |
| [RRPC\_FWSetFirewallRule2\_24](#Section_f6e7c7a0ffc74f72a20a0b57e784bd09) | This method requests the server to modify the specified firewall rule in the policy contained in the policy store that is referenced by the handle specified in the *hPolicyStore* parameter.  Opnum: 71 |
| [RRPC\_FWEnumFirewallRules2\_24](#Section_ee81af2bc01c4cf3812ebab6c89bd6ed) | This method requests the server to return all the firewall rules contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all the firewall rule objects.  Opnum: 72 |
| [RRPC\_FWQueryFirewallRules2\_24](#Section_0c82d80e54e34daf9cff4a92a54eaa84) | This method requests the server to return all the firewall rules matching the specified query object that are contained in the store that is referenced by the *hPolicyStore* handle.  Opnum: 73 |
| [RRPC\_FWAddFirewallRule2\_25](#Section_fb298ceac5fe4609b71ce0221dde97c1) | This method requests the server to add the specified firewall rule in the policy contained in the policy store that is referenced by the handle specified in the *hPolicyStore* parameter.  Opnum: 74 |
| [RRPC\_FWSetFirewallRule2\_25](#Section_7d89953750424032bbdc50cec6656c2c) | This method requests the server to modify the specified firewall rule in the policy contained in the policy store that is referenced by the handle specified in the *hPolicyStore* parameter.  Opnum: 75 |
| [RRPC\_FWEnumFirewallRules2\_25](#Section_17e6234cbd8e48439a2f884801ceddc9) | This method requests the server to return all the firewall rules contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all the firewall rule objects.  Opnum: 76 |
| [RRPC\_FWQueryFirewallRules2\_25](#Section_d6e826512e294f9999f13e2bd7cb93bd) | This method requests the server to return all the firewall rules matching the specified query object that are contained in the store that is referenced by the *hPolicyStore* handle.  Opnum: 77 |
| [RRPC\_FWAddFirewallRule2\_26](#Section_944db4f78e434b1b9060d9bd4ef8e7d1) | This method requests the server to add the specified firewall rule in the policy contained in the policy store that is referenced by the handle specified in the *hPolicyStore* parameter.  Opnum: 78 |
| [RRPC\_FWSetFirewallRule2\_26](#Section_d8b1914636d14a479d31bf0194f2b9da) | This method requests the server to modify the specified firewall rule in the policy contained in the policy store that is referenced by the handle specified in the *hPolicyStore* parameter.  Opnum: 79 |
| [RRPC\_FWEnumFirewallRules2\_26](#Section_cd34e45d4a254c928d34a07f3082c28e) | This method requests the server to return all the firewall rules contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all the firewall rule objects.  Opnum: 80 |
| [RRPC\_FWQueryFirewallRules2\_26](#Section_cf0aae4fbb2a42929bffd10b0b89f36e) | This method requests the server to return all the firewall rules matching the specified query object that are contained in the store that is referenced by the *hPolicyStore* handle.  Opnum: 81 |

#### RRPC\_FWOpenPolicyStore (Opnum 0)

The RRPC\_FWOpenPolicyStore method requests the server to open a specified policy store. The store can be opened for reading or for editing the firewall policy. The method also returns a handle to the opened store with which the client can then perform operations on this policy store. The server allocates a **PolicyStoreConnection** object to track the policy store type and the binary version associated with the handle.

1. unsigned long RRPC\_FWOpenPolicyStore(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] unsigned short BinaryVersion,
4. [in, range(FW\_STORE\_TYPE\_INVALID+1, FW\_STORE\_TYPE\_MAX-1)]
5. FW\_STORE\_TYPE StoreType,
6. [in, range(FW\_POLICY\_ACCESS\_RIGHT\_INVALID+1, FW\_POLICY\_ACCESS\_RIGHT\_MAX-1)]
7. FW\_POLICY\_ACCESS\_RIGHT AccessRight,
8. [in] unsigned long dwFlags,
9. [out] PFW\_POLICY\_STORE\_HANDLE phPolicyStore
10. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**BinaryVersion:** This parameter specifies the RPC interface binary version. This implies versions of the methods and versions of the structures. This value MUST be a valid Protocol Version (see section [1.7](#Section_6b14b8f991ab4f09873f9e2334196cdc)). See section 1.7 for capability negotiation based on the BinaryVersion.

**StoreType:** This parameter specifies the policy store type that the client wants to open.

**AccessRight:** This parameter specifies the read or read/write access rights that the client is requesting on the store.

**dwFlags:** This parameter is not used. The server MUST ignore this parameter. The client SHOULD pass a value of zero.

**phPolicyStore:** This is an output parameter that provides a pointer to an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. If successful, this parameter contains a handle to the opened store.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90).

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWClosePolicyStore (Opnum 1)

The RRPC\_FWClosePolicyStore method receives an opened store handle, closes it, and deallocates the corresponding **PolicyStoreConnection** object.

1. unsigned long RRPC\_FWClosePolicyStore(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in, out] PFW\_POLICY\_STORE\_HANDLE phPolicyStore
4. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**phPolicyStore:** This is an input and output parameter that provides a pointer to an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method, which the client intends to stop using and close.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90).

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWRestoreDefaults (Opnum 2)

The RRPC\_FWRestoreDefaults method replaces the contents of **LocalStore** with the contents of **DefaultsStore**.

1. unsigned long RRPC\_FWRestoreDefaults(
2. [in] FW\_CONN\_HANDLE rpcConnHandle
3. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90).

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST first validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method. Next, the server MUST replace the contents of **LocalStore** with the contents of **DefaultsStore**. The server then MUST merge the new contents of **LocalStore** with the existing contents of the **GroupPolicyRSoPStore** (as described in section [3.1.1](#Section_43507d538955416db913dfb27dc76b17)) and store the result in **DynamicStore**. Finally, the server MUST invoke the abstract interface [SetEffectiveFirewallPolicy (section 3.1.6.6)](#Section_0b159c9216fa42a69a33b68f0e040a98) with the contents of **DynamicStore**.

#### RRPC\_FWGetGlobalConfig (Opnum 3)

The RRPC\_FWGetGlobalConfig method retrieves the value of a global policy configuration option. The client specifies to the server from what store this value MUST be retrieved and in what specific configuration option it is interested.

1. unsigned long RRPC\_FWGetGlobalConfig(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] unsigned short BinaryVersion,
4. [in] FW\_STORE\_TYPE StoreType,
5. [in, range(FW\_GLOBAL\_CONFIG\_INVALID+1, FW\_GLOBAL\_CONFIG\_MAX-1)]
6. FW\_GLOBAL\_CONFIG configID,
7. [in] unsigned long dwFlags,
8. [in, out, unique, size\_is(cbData), length\_is(\*pcbTransmittedLen)]
9. unsigned char\* pBuffer,
10. [in] unsigned long cbData,
11. [in, out] unsigned long\* pcbTransmittedLen,
12. [out] unsigned long\* pcbRequired
13. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**BinaryVersion:** This parameter specifies the RPC interface binary version. This implies versions of the methods and versions of the structures.

**StoreType:** This parameter specifies the policy store from which the client wants to retrieve the configuration option value.

**configID:** This parameter specifies the specific global policy configuration option the client is interested in retrieving.

**dwFlags:** This parameter is a combination of flags from the [FW\_CONFIG\_FLAGS](#Section_eb0dce3b08d34c32b9fec8bdaf8e256a) enumeration, which modifies the behavior of this method, as specified in the definition of the enumeration.

**pBuffer:** This is an input/output parameter. This parameter is a pointer to the buffer that the client provides to contain the value of the profile configuration option being requested.

**cbData:** This parameter is the size of the buffer that the *pBuffer* parameter points to.

**pcbTransmittedLen:** This is a pointer to an input and output parameter that specifies the length of the transmitted data within the buffer.

**pcbRequired:** This is a pointer to an output parameter that specifies the required minimum buffer size in octets in order for the method to be able to return the configuration value. This output parameter is nonzero only if the buffer (pointed to by *pBuffer* and whose size is *cbData*) was not big enough to contain the value.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specific configuration option is not found within the policy. This means that it is not configured. If the option is not configured in any other store, the firewall uses a default value. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The store type specified does not support this method. |
| 0x000000EA  ERROR\_MORE\_DATA | The buffer is not big enough to hold the configuration option value. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect, or is required and not specified. This error can be returned because:   * The specific configuration option is not meant to be available in the specified store. * The specified configuration option is not defined. * One of the required values is not specified. * The buffer size is not enough to hold the specific value. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWSetGlobalConfig (Opnum 4)

The RRPC\_FWSetGlobalConfig method modifies the value of a global policy configuration option. The client specifies to the server in what store this value MUST be written and what specific configuration option it is interested in modifying.

1. unsigned long RRPC\_FWSetGlobalConfig(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] unsigned short BinaryVersion,
4. [in] FW\_STORE\_TYPE StoreType,
5. [in, range(FW\_GLOBAL\_CONFIG\_INVALID+1, FW\_GLOBAL\_CONFIG\_MAX-1)]
6. FW\_GLOBAL\_CONFIG configID,
7. [in, unique, size\_is(dwBufSize)]
8. unsigned char\* lpBuffer,
9. [in, range(0, 10\*1024)] unsigned long dwBufSize
10. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**BinaryVersion:** This parameter specifies the RPC interface binary version. This implies versions of the methods and versions of the structures.

**StoreType:** This parameter specifies the policy store in which the client wants to modify this configuration option.

**configID:** This parameter specifies the specific global policy configuration option the client wants to modify.

**lpBuffer:** This is an input parameter. This parameter is a pointer to the buffer that the client provides containing the value to write on the configuration option specified. If the buffer is NULL, this method deletes the configuration option.

**dwBufSize:** This parameter is the size of the buffer to which the *lpBuffer* parameter points.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The store type specified does not support this method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect, or is required and not specified. This error can be returned because:   * The specific configuration option is not meant to be available in the specified store. * The specified configuration option is not defined. * One of the required values is not specified. * The buffer is null but *dwBufSize* says otherwise. * The buffer size is not enough to hold the specific value. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method performs a merge operation of the resultant configuration values, as defined in section [3.1.3](#Section_e8924ac5aa4a41d1bf654f46b3d399aa). It then determines what modifications are necessary on the rule objects to make sure the policy is enforced.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWAddFirewallRule (Opnum 5)

The RRPC\_FWAddFirewallRule method requests the server to add the specified firewall rule in the policy contained in the policy store that is referenced by the handle specified in the *hPolicyStore* parameter.

1. ULONG RRPC\_FWAddFirewallRule(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] PFW\_RULE2\_0 pRule
5. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle that is successfully opened by using the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pRule:** This parameter represents the firewall rule that the client wants to add to the store. The rule MUST be a valid rule, as specified in the definition of the [FW\_RULE2\_0](#Section_af151922b0924e5c9b694dca6b6d2ffc) data type.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x000000B7  ERROR\_ALREADY\_EXISTS | The specified rule has a rule ID that already exists in the specified store. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | A parameter of this method is incorrect, or is required and not specified. This error can be returned because:   * The *pRule* object did not pass the firewall rule validations that are specified in the definition of the [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b) data type. * One of the required values is not specified. * A policy store does not support rules with profile conditions other than ALL profiles. * The **wszLocalApplication** field of the rule contains a string that was determined to be an invalid path.[<27>](#Appendix_A_27" \o "Product behavior note 27) |

**Exceptions Thrown**: No exceptions are thrown except those that are thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method adds a firewall rule to the firewall linked list of the memory representation of the store being modified. It also writes through and saves the rule in disk. If called on an online store, the firewall rule is also enforced.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWSetFirewallRule (Opnum 6)

The RRPC\_FWSetFirewallRule method requests the server to modify the specified firewall rule in the policy contained in the policy store that is referenced by the handle specified in the *hPolicyStore* parameter.

1. ULONG RRPC\_FWSetFirewallRule(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] PFW\_RULE2\_0 pRule
5. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle that is successfully opened by using the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pRule:** This parameter represents the firewall rule that the client wants to modify in the store. The rule MUST be a valid rule, as specified in the definition of the [FW\_RULE2\_0](#Section_af151922b0924e5c9b694dca6b6d2ffc) data type.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following are common.

| Return value/code | Description |
| --- | --- |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specified rule that is referenced by the **wszRuleID** member string of the [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b) data type is not found in the policy store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | A parameter of this method is incorrect, or is required and not specified. This error can be returned because:   * The *pRule* object did not pass the firewall rule validations that are specified in the definition of the FW\_RULE data type. * One of the required values is not specified. * A policy store does not support rules that have profile conditions other than ALL profiles. * The **wszLocalApplication** field of the rule contains a string that was determined to be an invalid path. |

**Exceptions Thrown**: No exceptions are thrown except those that are thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWDeleteFirewallRule (Opnum 7)

The RRPC\_FWDeleteFirewallRule method requests the server to delete the specified firewall rule in the policy contained in the policy store referenced by the handle specified in the *hPolicyStore* parameter.

1. unsigned long RRPC\_FWDeleteFirewallRule(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in, string, ref] const wchar\_t\* wszRuleID
5. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**wszRuleID:** This parameter is the pointer to a string that is the ID of the firewall rule the client wants to delete from the specified store.

This ID can be obtained by enumerating firewall rules using RRPC\_FWEnumFirewallRules (Opnum 9) where the ID is returned in the [FW\_RULE2\_0](#Section_af151922b0924e5c9b694dca6b6d2ffc) structure.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following are common.

| Return value/code | Description |
| --- | --- |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specified rule referenced by the **wszRuleID** member string of the [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b) data type is not found in the policy store. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method deletes a firewall rule already stored in the firewall linked list of the memory representation of the store being modified. It uses this list to determine if the rule exists or not. It also writes through and deletes the rule from disk. If called on an online store, the removal of the firewall rule is also enforced.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWDeleteAllFirewallRules (Opnum 8)

The RRPC\_FWDeleteAllFirewallRules method deletes all firewall rules in the firewall linked list of the memory representation of the store being modified. It also writes through and deletes all rules from the disk representation. If called on an online store, no firewall rules are enforced after the method returns.

1. unsigned long RRPC\_FWDeleteAllFirewallRules(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore
4. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following are common.

| Return value/code | Description |
| --- | --- |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWEnumFirewallRules (Opnum 9)

The RRPC\_FWEnumFirewallRules method requests the server to return all the firewall rules contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all the firewall rule objects.

1. ULONG RRPC\_FWEnumFirewallRules(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] ULONG dwFilteredByStatus,
5. [in] ULONG dwProfileFilter,
6. [in] USHORT wFlags,
7. [out, ref] ULONG\* pdwNumRules,
8. [out] PFW\_RULE2\_0\* ppRules
9. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle that is successfully opened by using the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read or read/write access rights.

**dwFilteredByStatus:** This parameter is a combination of flags from the [FW\_RULE\_STATUS\_CLASS](#Section_81877aa634c74664adf4ff8d5e0c0bc6) enumeration. This method uses this bitmask to determine which rules will be returned. Rules that contain a status code from the class specified by this parameter will be returned in the linked list.

**dwProfileFilter:** This parameter is a combination of flags from the [FW\_PROFILE\_TYPE](#Section_7704e238174d4a5eb8095f3787dd8acc) enumeration. This method also uses this parameter to determine which rules will be returned. Rules that contain a profile specified by this parameter will be returned in the linked list.

**wFlags:** This parameter is a combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumRules:** This output parameter, if successful, MUST be equal to the number of rules returned.

**ppRules:** This output parameter, if successful, contains a linked list of [FW\_RULE2\_0](#Section_af151922b0924e5c9b694dca6b6d2ffc) data types.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *dwProfileFilter* parameter contains profiles that are not valid. |

**Exceptions Thrown**: No exceptions are thrown except those that are thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWGetConfig (Opnum 10)

The RRPC\_FWGetConfig method retrieves the value of a profile configuration option. The client specifies to the server from what store and profile this value MUST be retrieved and in what specific configuration option it is interested.

1. unsigned long RRPC\_FWGetConfig(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in, range(FW\_PROFILE\_CONFIG\_ENABLE\_FW, FW\_PROFILE\_CONFIG\_MAX-1)]
5. FW\_PROFILE\_CONFIG configID,
6. [in] FW\_PROFILE\_TYPE Profile,
7. [in] unsigned long dwFlags,
8. [in, out, unique, size\_is(cbData), length\_is(\*pcbTransmittedLen)]
9. unsigned char\* pBuffer,
10. [in] unsigned long cbData,
11. [in, out] unsigned long\* pcbTransmittedLen,
12. [out] unsigned long\* pcbRequired
13. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**configID:** This parameter specifies the specific profile configuration option the client is interested in retrieving.

**Profile:** This parameter specifies from which specific profile this value MUST be retrieved.

**dwFlags:** This parameter is a combination of flags from the [FW\_CONFIG\_FLAGS](#Section_eb0dce3b08d34c32b9fec8bdaf8e256a) enumeration, which modifies the behavior of this method, as specified in the definition of the enumeration.

**pBuffer:** This is an input/output parameter. This parameter is a pointer to the buffer that the client provides to contain the value of the profile configuration option being requested.

**cbData:** This parameter is the size of the buffer that the *pBuffer* parameter points to.

**pcbTransmittedLen:** This is a pointer to an input and output parameter that specifies the length of the transmitted data within the buffer.

**pcbRequired:** This is a pointer to an output parameter that specifies the required minimum buffer size in octets for the method to be able to return the configuration value. This output parameter is nonzero only if the buffer (pointed to by *pBuffer* and whose size is *cbData*) was not big enough to contain the value.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specific configuration option is not found within the policy. This means that it is not configured. If the option is not configured in any other store, the firewall uses a default value. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The method does not support the specified combination of parameters. This can be because:   * The store type specified does not support this method. * The configuration option is not supported in this store. * The *Profile* parameter contains a combination of profiles (instead of a single profile) or an unknown profile. |
| 0x000000EA  ERROR\_MORE\_DATA | The buffer is not big enough to hold the configuration option value. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect, or is required and not specified. This error can be returned because:   * The specified configuration option is not defined. * One of the required values is not specified. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWSetConfig (Opnum 11)

The RRPC\_FWSetConfig method modifies the value of a profile configuration option. The client specifies to the server in what store and profile this value MUST be written and what specific configuration option it is interested in modifying.

1. unsigned long RRPC\_FWSetConfig(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in, range(FW\_PROFILE\_CONFIG\_ENABLE\_FW, FW\_PROFILE\_CONFIG\_MAX-1)]
5. FW\_PROFILE\_CONFIG configID,
6. [in] FW\_PROFILE\_TYPE Profile,
7. [in, switch\_is(configID)] FW\_PROFILE\_CONFIG\_VALUE pConfig,
8. [in, range(0, 10\*1024)] unsigned long dwBufSize
9. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**configID:** This parameter specifies the specific profile configuration option the client wants to modify.

**Profile:** This parameter specifies in which specific profile this value MUST be written.

**pConfig:** This is an input parameter. This parameter is a pointer to the buffer that the client provides containing the value to write on the configuration option specified. If the buffer is NULL, this method deletes the configuration option. The buffer is of type [FW\_PROFILE\_CONFIG\_VALUE](#Section_f5af7cf8f9484265b50e80fd73edf3ad).

**dwBufSize:** This parameter is the size of the buffer that the *pConfig* parameter points to.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The method does not support the specified combination of parameters. This can be because:   * The store type specified does not support this method. * The Profile parameter contains a combination of profiles (instead of a single profile) or an unknown profile. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect, or is required and not specified. This error can be returned because:   * The specific configuration option is not meant to be available in the specified store. * The specified configuration option is not defined. * The size of the buffer does not match the size of the type of the configuration value. * The buffer is null but *dwBufSize* says otherwise. * The caller wants to set a LOG\_MAX\_FILE\_SIZE that is not within the valid values [min, max]. * The default action configuration value specifies a value that maps to neither allow nor block. * The LOG\_FILE\_PATH configuration value contains the following invalid characters: /,\*,?,",<,>,|. |

**Exceptions Thrown:** No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [MS-RPCE]. If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method performs a merge operation of the resultant configuration values, as defined in section [3.1.3](#Section_e8924ac5aa4a41d1bf654f46b3d399aa). It then determines what modifications are necessary on the rule objects (for example, remove rule enforcement if firewall is off) to make sure the policy is enforced.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWAddConnectionSecurityRule (Opnum 12)

The RRPC\_FWAddConnectionSecurityRule method requests the server to add the connection security rule in the policy contained in the policy store that is referenced by the specified opened policy store handle.

1. ULONG RRPC\_FWAddConnectionSecurityRule(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in] PFW\_CS\_RULE2\_0 pRule
5. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle that is successfully opened by using the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pRule:** This parameter represents the connection security rule that the client wants to add to the store. The rule MUST be a valid rule, as specified in the definition of the [FW\_CS\_RULE2\_0](#Section_18277e4c628d44239fe3ba98601ba50d) data type.

**Return Values:** This method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x000000B7  ERROR\_ALREADY\_EXISTS | The specified rule has a rule ID that already exists in the specified store. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicy* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | A parameter of this method is incorrect, or is required and not specified. This error can be returned because:   * The *pRule* object did not pass the connection security rule validations specified in the definition of the [FW\_CS\_RULE](#Section_0d0641105f2e4b68aa63032c6cd5e4c6) data type. * The rule has a phase 2 crypto set that specified FW\_CRYPTO\_PRPTOCOL\_AUTH\_NO\_ENCAP (see section [2.2.68](#Section_d97bac3603e842159984f2fbddd66be0)), and it is a tunnel mode rule, or it also has an **AuthSet** structure that specifies a preshared key auth method. * A required value is not specified. |

**Exceptions Thrown**: No exceptions are thrown except those that are thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method adds a connection security rule in the connection security link list of the memory representation of the store being modified. It also writes through and saves the rule to disk. If called on an online store, the connection security rule is also enforced.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWSetConnectionSecurityRule (Opnum 13)

The RRPC\_FWSetConnectionSecurityRule method requests the server to modify the specified connection security rule in the policy contained in the policy store that is referenced by the handle specified in the *hPolicy* parameter.

1. ULONG RRPC\_FWSetConnectionSecurityRule(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in] PFW\_CS\_RULE2\_0 pRule
5. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle that is successfully opened by using the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pRule:** This parameter represents the connection security rule that the client wants to modify in the store. The rule MUST be a valid rule, as specified in the definition of the [FW\_CS\_RULE2\_0](#Section_18277e4c628d44239fe3ba98601ba50d) data type.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicy* handle was not opened with read/write access rights. This error is also returned if the client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specified rule that is referenced by the *wszRuleID* member string of the [FW\_CS\_RULE](#Section_0d0641105f2e4b68aa63032c6cd5e4c6) data type is not found in the policy store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | A parameter of this method is incorrect, or is required and not specified. This error can be returned because:   * The *pRule* object did not pass the connection security rule validations that are specified in the definition of the FW\_CS\_RULE data type. * The rule has a phase 2 crypto set that specified FW\_CRYPTO\_PRPTOCOL\_AUTH\_NO\_ENCAP (see section [2.2.68](#Section_d97bac3603e842159984f2fbddd66be0)), and either it is a tunnel mode rule or it has an AuthSet that specifies a preshared key auth method. * A required value is not specified. |

**Exceptions Thrown**: No exceptions are thrown except those that are thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method modifies a connection security rule already stored in the connection security linked list of the memory representation of the store being modified. It uses this list to determine whether the rule exists. It also writes through and saves the rule in disk. If called on an online store, the connection security rule modifications are also enforced.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWDeleteConnectionSecurityRule (Opnum 14)

The RRPC\_FWDeleteConnectionSecurityRule method requests the server to delete the specified connection security rule in the policy contained in the policy store referenced by the handle specified in the *hPolicy* parameter.

1. unsigned long RRPC\_FWDeleteConnectionSecurityRule(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in, string, ref] wchar\_t\* pRuleId
5. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pRuleId:** This parameter is the pointer to a string that is the ID of the connection security rule the client wants to delete from the specified store.

This ID can be obtained by enumerating connection security rules using RRPC\_FWEnumConnectionSecurityRules (Opnum 16) where the ID is returned in the [FW\_CS\_RULE2\_0](#Section_18277e4c628d44239fe3ba98601ba50d) structure.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicy* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specified rule referenced by the *pRuleID* member string is not found in the policy store. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method deletes a connection security rule already stored in the connection security linked list of the memory representation of the store being modified. It uses this list to determine if the rule exists or not. It also writes through and deletes the rule from disk. If called on an online store, the removal of the connection security rule is also enforced.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWDeleteAllConnectionSecurityRules (Opnum 15)

The RRPC\_FWDeleteAllConnectionSecurityRules method requests the server to delete all the connection security rules in the policy contained in the policy store referenced by the handle specified in the *hPolicy* parameter.

1. unsigned long RRPC\_FWDeleteAllConnectionSecurityRules(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy
4. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicy* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method deletes all connection security rules in the connection security linked list of the memory representation of the store being modified. It also writes through and deletes all rules from the disk representation. If called on an online store, no connection security rules are enforced after the method returns.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWEnumConnectionSecurityRules (Opnum 16)

The RRPC\_FWEnumConnectionSecurityRules method requests the server to return all the connection security rules contained in the store that is referenced by the *hPolicy* handle. The method returns a linked list of all the connection security rule objects.

1. ULONG RRPC\_FWEnumConnectionSecurityRules(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in] ULONG dwFilteredByStatus,
5. [in] ULONG dwProfileFilter,
6. [in] USHORT wFlags,
7. [out, ref] ULONG\* pdwNumRules,
8. [out] PFW\_CS\_RULE2\_0\* ppRules
9. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle that is successfully opened by using the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read or read/write access rights.

**dwFilteredByStatus:** This parameter is a combination of flags from the [FW\_RULE\_STATUS\_CLASS](#Section_81877aa634c74664adf4ff8d5e0c0bc6) enumeration. This method uses this bitmask to determine which rules will be returned. Rules that contain a status code from the class that is specified by this parameter will be returned in the linked list.

**dwProfileFilter:** This parameter is a combination of flags from the [FW\_PROFILE\_TYPE](#Section_7704e238174d4a5eb8095f3787dd8acc) enumeration. This method also uses this parameter to determine which rules will be returned. Rules that contain a profile that is specified by this parameter will be returned in the linked list.

**wFlags:** This parameter is a combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumRules:** This output parameter, if successful, MUST be equal to the number of rules returned.

**ppRules:** This output parameter, if successful, contains a linked list of [FW\_CS\_RULE2\_0](#Section_18277e4c628d44239fe3ba98601ba50d) data types.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *dwProfileFilter* parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown except those that are thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWAddAuthenticationSet (Opnum 17)

The RRPC\_FWAddAuthenticationSet method requests the server to add the authentication set in the policy contained in the policy store referenced by the handle specified in the *hPolicy* parameter.

1. unsigned long RRPC\_FWAddAuthenticationSet(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in] PFW\_AUTH\_SET2\_10 pAuth
5. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pAuth:** This parameter represents the authentication set the client wants to add to the store. The set MUST be valid, as specified in the definition of the [FW\_AUTH\_SET2\_10](#Section_c1acee0a747a4482b1590c96ca57fe29) data type.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x000000B7  ERROR\_ALREADY\_EXISTS | The specified set has a set ID that already exists in the specified store. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicy* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect, or is required and not specified. This error can be returned because:   * The *pAuth* object did not pass the authentication set validations specified in the definition of the [FW\_AUTH\_SET](#Section_bbec7ab70a1d49a49c8ea4784bbe6f16) data type. * One of the required values is not specified. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method adds an authentication set in the authentication linked list of the memory representation of the store being modified. It also writes through and saves the set in disk. If called on an online store and the set is a primary set, the method enumerates the connection security rule list and reapplies each rule referencing this primary set to complete the enforcement of the policy.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWSetAuthenticationSet (Opnum 18)

The RRPC\_FWSetAuthenticationSet method requests the server to modify the specified authentication set in the policy contained in the policy store referenced by the handle specified in the *hPolicy* parameter.

1. unsigned long RRPC\_FWSetAuthenticationSet(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in] PFW\_AUTH\_SET2\_10 pAuth
5. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pAuth:** This parameter represents the authentication set the client wants to modify in the store. The set MUST be valid, as specified in the definition of the [FW\_AUTH\_SET2\_10](#Section_c1acee0a747a4482b1590c96ca57fe29) data type.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicy* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specified set referenced by the **wszSetID** member string of the [FW\_AUTH\_SET](#Section_bbec7ab70a1d49a49c8ea4784bbe6f16) data type is not found in the policy store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect, or is required and not specified. This error can be returned because:   * The *pAuth* object did not pass the authentication set validations specified in the definition of the FW\_AUTH\_SET data type. * One of the required values is not specified. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method modifies an authentication set in the authentication linked list of the memory representation of the store being modified. It also writes through and saves the set in disk. If called on an online store, the method enumerates the connection security rules list and reapplies each rule referencing this primary set to complete the enforcement of the policy.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWDeleteAuthenticationSet (Opnum 19)

The RRPC\_FWDeleteAuthenticationSet method requests the server to delete the specified authentication set in the policy contained in the policy store referenced by the handle specified in the *hPolicy* parameter.

1. unsigned long RRPC\_FWDeleteAuthenticationSet(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)]
5. FW\_IPSEC\_PHASE IpSecPhase,
6. [in, string, ref] const wchar\_t\* wszSetId
7. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**IpSecPhase:** This parameter specifies the [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) negotiation phase type this set is used in.

**wszSetId:** This parameter is the pointer to a string that is the ID of the authentication set the client wants to delete from the specified store.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000962  ERROR\_ACTIVE\_CONNECTIONS | The specified set is still referenced by connection security rules. This failure happens only when the set is not a primary set. There is always a primary set to use, either from other stores or a hard-coded one. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicy* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specified rule referenced by the **wszSetID** string is not found in the policy store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The specified IPsec phase is not a valid one. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method deletes an authentication set in the authentication linked list of the memory representation of the store being modified. It also writes through and saves the set in disk. If called on an online store, and the set is not a primary set, the method does not delete the specified set if any connection rule references this set.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWDeleteAllAuthenticationSets (Opnum 20)

The RRPC\_FWDeleteAllAuthenticationSets method requests the server to delete all the authentication sets of a specific [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) phase in the policy contained in the policy store referenced by the handle specified in the *hPolicy* parameter.

1. unsigned long RRPC\_FWDeleteAllAuthenticationSets(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)]
5. FW\_IPSEC\_PHASE IpSecPhase
6. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**IpSecPhase:** This parameter specifies the IPsec negotiation phase type in which this set is used.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000962  ERROR\_ACTIVE\_CONNECTIONS | The specified set is still referenced by connection security rules. This failure happens only when the set is not a primary set. There is always a primary set to use, either from other stores or a hard-coded one. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicy* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specified rule referenced by the **wszSetID** string is not found in the policy store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The specified IPsec phase is not a valid one. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method deletes all the authentication sets in the authentication linked list of the memory representation of the store being modified. It also writes through and deletes the sets from disk. If called on an online store, the method does not delete the sets if any nonprimary set is referenced by a connection security rule.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWEnumAuthenticationSets (Opnum 21)

The RRPC\_FWEnumAuthenticationSets method requests the server to return all the authentication sets of the specified [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) phase contained in the store referenced by the *hPolicy* handle. The method returns a linked list of these objects.

1. unsigned long RRPC\_FWEnumAuthenticationSets(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)]
5. FW\_IPSEC\_PHASE IpSecPhase,
6. [in] unsigned long dwFilteredByStatus,
7. [in] unsigned short wFlags,
8. [out] unsigned long\* pdwNumAuthSets,
9. [out] PFW\_AUTH\_SET2\_10\* ppAuth
10. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read or read/write access rights.

**IpSecPhase:** This parameter specifies the specific IPsec negotiation phase to which this set applies.

**dwFilteredByStatus:** This parameter is a combination of flags from the [FW\_RULE\_STATUS\_CLASS](#Section_81877aa634c74664adf4ff8d5e0c0bc6) enumeration. This method uses this bitmask to determine which rules will be returned. Sets that contain a status code of the class specified by this parameter will be returned in the linked list.

**wFlags:** This parameter is a combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) that modifies the behavior of the method and performs operations on the sets before returning them in the linked list.

**pdwNumAuthSets:** This is an output parameter that on success MUST be equal to the number of sets returned.

**ppAuth:** This is an output parameter that on success contains a linked list of [FW\_AUTH\_SET2\_10](#Section_c1acee0a747a4482b1590c96ca57fe29) data types.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect, or is required and not specified. This error can be returned because:   * The *IpSecPhase* parameter specifies an invalid IPsec negotiation phase. * One of the required values is not specified. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

When this method is called, the server looks for the binary version of the client, which was associated with the *hPolicy* handle when the client sent the [RRPC\_FWOpenPolicyStore()](#Section_2157e39a1bf04e0cabae0811fd918b11) call. The server compares this binary version parameter with the schema version it supports. If the server has a schema version of 0x0201 and the client passed a 0x0200 binary version, then the server removes all values that are not valid for a [FW\_AUTH\_SET (section 2.2.64)](#Section_bbec7ab70a1d49a49c8ea4784bbe6f16) structure that has a 0x0200 schema version. If the removed value was present on one or more suites of the set, the server removes those suites as a whole, leaving the remaining suites intact. For each set that had a value removed, the server sets a FW\_RULE\_STATUS\_PARTIALLY\_IGNORED value on the **Status** field of the set. Then the client receives authentication sets with values that correspond to the correct schema version, but the client recognizes that the information it has about the sets is potentially incomplete.

#### RRPC\_FWAddCryptoSet (Opnum 22)

The RRPC\_FWAddCryptoSet method adds a cryptographic set in the cryptographic linked list of the memory representation of the store being modified. It also writes through and saves the set to the disk. If called on an online store, and the set is a primary set, the method enumerates the connection security rule list and reapplies each rule referencing this primary set to complete the enforcement of the policy.

The server MUST determine whether the local computer is operating in [**common criteria mode**](#gt_52549a11-2432-4a5c-966f-5f8a32de9162) by invoking the abstract interface [IsComputerInCommonCriteriaMode (section 3.1.6.5)](#Section_6da7ac30956b42cbb7dcc0af8b9db707). If the local computer is operating in common criteria mode, the server MUST fail the operation and return an error of ERROR\_ACCESS\_DENIED (5). Otherwise, the server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

1. unsigned long RRPC\_FWAddCryptoSet(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in] PFW\_CRYPTO\_SET pCrypto
5. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pCrypto:** This parameter represents the cryptographic set the client wants to add to the store. The set MUST be valid, as specified in the definition of the [FW\_CRYPTO\_SET](#Section_a468fe9e113b4155a63d0db3aac12619) data type.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x000000B7  ERROR\_ALREADY\_EXISTS | The specified rule has a rule ID that already exists in the specified store. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicy* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect, or is required and not specified. This error can be returned because:   * The *pCrypto* object did not pass the cryptographic set validations specified in the definition of the FW\_CRYPTO\_SET data type. * One of the required values is not specified. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

#### RRPC\_FWSetCryptoSet (Opnum 23)

The RRPC\_FWSetCryptoSet method requests the server to modify the specified cryptographic set in the policy contained in the policy store referenced by the handle specified in the *hPolicy* parameter.

1. unsigned long RRPC\_FWSetCryptoSet(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in] PFW\_CRYPTO\_SET pCrypto
5. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pCrypto:** This parameter represents the cryptographic set the client wants to modify in the store. The set MUST be valid, as specified in the definition of the [FW\_CRYPTO\_SET](#Section_a468fe9e113b4155a63d0db3aac12619) data type.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicy* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specified set referenced by the **wszSetID** member string of the FW\_CRYPTO\_SET data type is not found in the policy store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect, or is required and not specified. This error can be returned because:   * The *pCrypto* object did not pass the cryptographic set validations specified in the definition of the FW\_CRYPTO\_SET data type. * One of the required values is not specified. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method modifies a cryptographic set in the cryptographic linked list of the memory representation of the store being modified. It also writes through and saves the set to the disk. If called on an online store, the method enumerates the connection security rules list and reapplies each rule referencing this primary set to complete the enforcement of the policy.

The server MUST determine whether the local computer is operating in [**common criteria mode**](#gt_52549a11-2432-4a5c-966f-5f8a32de9162) by invoking the abstract interface [IsComputerInCommonCriteriaMode (section 3.1.6.5)](#Section_6da7ac30956b42cbb7dcc0af8b9db707). If the local computer is operating in common criteria mode, the server MUST fail the operation and return an error of ERROR\_ACCESS\_DENIED (5). Otherwise, the server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWDeleteCryptoSet (Opnum 24)

The RRPC\_FWDeleteCryptoSet method requests the server to delete the specified cryptographic set in the policy contained in the policy store that is referenced by the handle specified in the *hPolicy* parameter.

1. ULONG RRPC\_FWDeleteCryptoSet(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)]
5. FW\_IPSEC\_PHASE IpSecPhase,
6. [in, string, ref] const wchar\_t\* wszSetId
7. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle that is successfully opened by using the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**IpSecPhase:** This parameter specifies the [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) negotiation phase type in which this set is used.

**wszSetId:** This parameter is the pointer to a string that is the ID of the cryptographic set that the client wants to delete from the specified store.

This ID can be obtained by enumerating cryptographic sets using the RRPC\_FWEnumCryptoSets (Opnum 26) where the ID is returned in the [FW\_CRYPTO\_SET](#Section_a468fe9e113b4155a63d0db3aac12619) structure.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000962  ERROR\_ACTIVE\_CONNECTIONS | The specified set is still referenced by connection security or main mode rules. This failure happens only when the set is not a primary set. There is always a primary set to use, either from other stores or a hard-coded one. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicy* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specified rule that is referenced by the **wszSetID** string is not found in the policy store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The specified IPsec phase is not a valid one. |

**Exceptions Thrown**: No exceptions are thrown except those that are thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method deletes a cryptographic set in the cryptographic linked list of the memory representation of the store being modified. It also writes through and saves the set to disk. If called on an online store and the set is not a primary set, the method does not delete the specified set if any connection rule references this set.

The server MUST determine whether the local computer is operating in [**common criteria mode**](#gt_52549a11-2432-4a5c-966f-5f8a32de9162) by invoking the abstract interface [IsComputerInCommonCriteriaMode (section 3.1.6.5)](#Section_6da7ac30956b42cbb7dcc0af8b9db707). If the local computer is operating in common criteria mode, the server MUST fail the operation and return an error of ERROR\_ACCESS\_DENIED (5). Otherwise, the server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWDeleteAllCryptoSets (Opnum 25)

The RRPC\_FWDeleteAllCryptoSets method requests the server to delete all the cryptographic sets of a specific [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) phase in the policy contained in the policy store that is referenced by the handle specified in the *hPolicy* parameter.

1. unsigned long RRPC\_FWDeleteAllCryptoSets(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)]
5. FW\_IPSEC\_PHASE IpSecPhase
6. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle that is successfully opened by using the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**IpSecPhase:** This parameter specifies the IPsec negotiation phase type in which this set is used.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000962  ERROR\_ACTIVE\_CONNECTIONS | There are nonprimary sets still being referenced by connection security or main mode rules. There is always a primary set to use, either from other stores or a hard-coded one; therefore, this failure never occurs because of primary sets. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicy* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The specified IPsec phase is not a valid one. |

**Exceptions Thrown**: No exceptions are thrown except those that are thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method deletes all the cryptographic sets in the cryptographic linked list of the memory representation of the store being modified. It also writes through and deletes the sets from disk. If called on an online store, the method does not delete the sets if any nonprimary set is referenced by a connection security rule.

The server MUST determine whether the local computer is operating in [**common criteria mode**](#gt_52549a11-2432-4a5c-966f-5f8a32de9162) by invoking the abstract interface [IsComputerInCommonCriteriaMode (section 3.1.6.5)](#Section_6da7ac30956b42cbb7dcc0af8b9db707). If the local computer is operating in common criteria mode, the server MUST fail the operation and return an error of ERROR\_ACCESS\_DENIED (5). Otherwise, the server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWEnumCryptoSets (Opnum 26)

The RRPC\_FWEnumCryptoSets method requests the server to return all the cryptographic sets of the specified [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) phase contained in the store referenced by the *hPolicy* handle. The method returns a linked list of all these cryptographic objects.

1. unsigned long RRPC\_FWEnumCryptoSets(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)]
5. FW\_IPSEC\_PHASE IpSecPhase,
6. [in] unsigned long dwFilteredByStatus,
7. [in] unsigned short wFlags,
8. [out, ref] unsigned long\* pdwNumSets,
9. [out] PFW\_CRYPTO\_SET\* ppCryptoSets
10. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read or read/write access rights.

**IpSecPhase:** This parameter specifies the specific IPsec negotiation phase to which this set applies.

**dwFilteredByStatus:** This parameter is a combination of flags from the [FW\_RULE\_STATUS\_CLASS](#Section_81877aa634c74664adf4ff8d5e0c0bc6) enumeration. This method uses this bitmask to determine which rules will be returned. Sets that contain a status code of the class specified by matches to this parameter will be returned in the linked list.

**wFlags:** This parameter is a combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) that modifies the behavior of the method and performs operations on the sets before returning them in the linked list.

**pdwNumSets:** This is an output parameter that on success MUST be equal to the number of sets returned.

**ppCryptoSets:** This is an output parameter that on success contains a linked list of [FW\_CRYPTO\_SET](#Section_a468fe9e113b4155a63d0db3aac12619) data types.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect, or is required and not specified. This error can be returned because:   * The *IpSecPhase* parameter specifies an invalid IPsec negotiation phase. * One of the required values is not specified. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

When this method is called, the server looks for the binary version of the client, which was associated with the hPolicy handle when the client sent the [RRPC\_FWOpenPolicyStore()](#Section_2157e39a1bf04e0cabae0811fd918b11) call. The server compares this binary version parameter with the schema version that it supports. If the server has a schema version of 0x0201 and the client passed a 0x0200 binary version, the server removes all values that are not valid for a FW\_CRYPTO\_SET (section 2.2.73) structure that has a 0x0200 schema version. If the removed value was present on one or more suites of the set, the server removes those suites as a whole, leaving the remaining suites intact. For each set that had a value removed, the server sets a FW\_RULE\_STATUS\_PARTIALLY\_IGNORED value on the **Status** field of the set. The client then receives cryptographic sets with values that correspond to the correct schema version, but the client recognizes that the information it has about the sets is potentially incomplete.

#### RRPC\_FWEnumPhase1SAs (Opnum 27)

The RRPC\_FWEnumPhase1SAs method requests the server to return all the [**security associations**](#gt_67cbf867-7a49-41f3-a68f-37b5f9035acb) of the [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) first negotiation phase contained in the store referenced by the *hPolicy* handle. The method returns a linked list of all these security associations.

1. unsigned long RRPC\_FWEnumPhase1SAs(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in, unique] PFW\_ENDPOINTS pEndpoints,
5. [out, ref] unsigned long\* pdwNumSAs,
6. [out, size\_is(, \*pdwNumSAs)] PFW\_PHASE1\_SA\_DETAILS\* ppSAs
7. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. This handle MUST be of the FW\_STORE\_TYPE\_DYNAMIC store.

**pEndpoints:** This parameter is a pointer to an [FW\_ENDPOINTS](#Section_2318781e64bc475d997502cc9d56ef0b) data type that can hold the addresses of the destination and source host. These addresses are used to match the security associations that will be returned. If this parameter is NULL, the method returns all IPsec first-phase security associations.

**pdwNumSAs:** This is an output parameter that on success MUST be equal to the number of security associations returned.

**ppSAs:** This is an output parameter that on success contains a linked list of [FW\_PHASE1\_SA\_DETAILS](#Section_218b010dea7f4b14b728aa27dc418236) data types, each of which represents the first-phase security association.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The store handle is not of the dynamic store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect, or is required and not specified. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

When this method is called, the server looks for the binary version of the client, which was associated with the *hPolicy* handle when the client sent the [RRPC\_FWOpenPolicyStore](#Section_2157e39a1bf04e0cabae0811fd918b11) call. The server compares this binary version parameter with the schema version that it supports. If the server’s schema version is greater than the binary version passed by the client, the server removes all FW\_PHASE1\_SA\_DETAILS objects that contain values that are not valid for an [FW\_AUTH\_SET](#Section_bbec7ab70a1d49a49c8ea4784bbe6f16) (section 2.2.64) structure that has the schema version value passed by the client.

#### RRPC\_FWEnumPhase2SAs (Opnum 28)

The RRPC\_FWEnumPhase2SAs method requests the server to return all the [**security associations**](#gt_67cbf867-7a49-41f3-a68f-37b5f9035acb) of the IPsec second negotiation phase contained in the store referenced by the *hPolicy* handle. The method returns a linked list of all these security associations.

1. unsigned long RRPC\_FWEnumPhase2SAs(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in, unique] PFW\_ENDPOINTS pEndpoints,
5. [out, ref] unsigned long\* pdwNumSAs,
6. [out, size\_is(, \*pdwNumSAs)] PFW\_PHASE2\_SA\_DETAILS\* ppSAs
7. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. This handle MUST be of the FW\_STORE\_TYPE\_DYNAMIC store.

**pEndpoints:** This parameter is a pointer to an [FW\_ENDPOINTS](#Section_2318781e64bc475d997502cc9d56ef0b) data type that can hold the addresses of the destination and source host. These addresses are used to match the security associations that will be returned. If this parameter is NULL, the method will return all [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) second phase security associations. If an [**endpoint**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee) is empty (that is, equal to 0), the endpoint matches any address.

**pdwNumSAs:** This is an output parameter that on success MUST be equal to the number of security associations returned.

**ppSAs:** This is an output parameter that on success contains a linked list of [FW\_PHASE2\_SA\_DETAILS](#Section_e6091d0ae57846e08656113d1e1e360d) data types, each of which represents a second phase security association.

**Return Values:** The method returns 0 if successful; if failed, it returns a non-zero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The store handle is not of the dynamic store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect, or is required and not specified. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWDeletePhase1SAs (Opnum 29)

The RRPC\_FWDeletePhase1SAs method requests the server to delete all the [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) first negotiation phase [**security associations**](#gt_67cbf867-7a49-41f3-a68f-37b5f9035acb) that match the specified [**endpoints**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee).

1. unsigned long RRPC\_FWDeletePhase1SAs(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in, unique] PFW\_ENDPOINTS pEndpoints
5. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. This handle MUST be of the FW\_STORE\_TYPE\_DYNAMIC store.

**pEndpoints:** This parameter is a pointer to an [FW\_ENDPOINTS](#Section_2318781e64bc475d997502cc9d56ef0b) data type that can hold the addresses of the destination and source host. These addresses are used to match the security associations that will be deleted. If this parameter is NULL, the method deletes all IPsec first-phase security associations. If an endpoint is empty (that is, equal to 0), the endpoint matches any address.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The store handle is not of the dynamic store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect or is required and not specified. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWDeletePhase2SAs (Opnum 30)

The RRPC\_FWDeletePhase2SAs (Opnum 30) method requests the server to delete all the [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) second-negotiation-phase [**security associations**](#gt_67cbf867-7a49-41f3-a68f-37b5f9035acb) that match the specified [**endpoints**](#gt_b91c1e27-e8e0-499b-8c65-738006af72ee).

1. unsigned long RRPC\_FWDeletePhase2SAs(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in, unique] PFW\_ENDPOINTS pEndpoints
5. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. This handle MUST be of the FW\_STORE\_TYPE\_DYNAMIC store.

**pEndpoints:** This parameter is a pointer to an [FW\_ENDPOINTS](#Section_2318781e64bc475d997502cc9d56ef0b) data type that can hold the addresses of the destination and source host. These addresses are used to match the security associations that will be deleted. If this parameter is NULL, the method deletes all IPsec second-phase security associations. If an endpoint is empty (that is, equal to 0), the endpoint matches any address.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The store handle is not of the dynamic store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect, or is required and not specified. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWEnumProducts (Opnum 31)

The RRPC\_FWEnumProducts (Opnum 31) method requests the server to return all the registered third-party software components registered with the firewall and advanced security component. The only method supported is binary version 0x020A.

1. unsigned long RRPC\_FWEnumProducts(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [out] unsigned long\* pdwNumProducts,
5. [out, size\_is(,\*pdwNumProducts)]
6. PFW\_PRODUCT\* ppProducts
7. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. This handle MUST be of the [FW\_STORE\_TYPE\_DYNAMIC](#Section_37ebed958abf472c8b4b7a510a2a6baa) store.

**pdwNumProducts:** This is an output parameter that on success MUST be equal to the number of products returned.

**ppProducts:** An array of [FW\_PRODUCT](#Section_969d38731e424dc48559f5c0c9caa965) data types, representing the registration of third-party software components.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The store handle is not of the dynamic store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method either is incorrect or is required and not specified. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWAddMainModeRule (Opnum 32)

The RRPC\_FWAddMainModeRule (Opnum 32) method requests the server to add the main mode rule in the policy contained in the policy store referenced by the specified opened policy store handle. The only method supported is binary version 0x020A.

1. unsigned long RRPC\_FWAddMainModeRule(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in] PFW\_MM\_RULE pMMRule,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. This handle MUST be of the [FW\_STORE\_TYPE\_DYNAMIC](#Section_37ebed958abf472c8b4b7a510a2a6baa) store.

**pMMRule:** This parameter represents the main mode rule that the client adds in the store. The rule MUST be valid, as specified in the definition of the [FW\_MM\_RULE](#Section_4b3fc163fede434c90cb557992caa5da) data type.

**pStatus:** This is an output parameter that on return will have the status code of the rule.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x000000B7  ERROR\_ALREADY\_EXISTS | The specified rule has a rule ID that already exists in the specified store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method either is incorrect or is required and not specified. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWSetMainModeRule (Opnum 33)

The RRPC\_FWSetMainModeRule (Opnum 33) method requests the server to modify the specified main mode rule in the policy contained in the policy store referenced by the handle specified in the *hPolicy* parameter. The only method supported is binary version 0x020A.

1. unsigned long RRPC\_FWSetMainModeRule(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in] PFW\_MM\_RULE pMMRule,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. This handle MUST be of the [FW\_STORE\_TYPE\_DYNAMIC](#Section_37ebed958abf472c8b4b7a510a2a6baa) store.

**pMMRule:** This parameter represents the main mode rule the client modifies in the store. The rule MUST be valid, as specified in the definition of the [FW\_MM\_RULE](#Section_4b3fc163fede434c90cb557992caa5da) data type.

**pStatus:** This is an output parameter that on return will have the status code of the rule.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specified set referenced by the **wszRuleID** member STRING of the FW\_MM\_RULE data type is not found in the policy store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method either is incorrect or is required and not specified. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWDeleteMainModeRule (Opnum 34)

The RRPC\_FWDeleteMainModeRule (Opnum 34) method requests the server to delete the specified main mode rule in the policy contained in the policy store referenced by the handle specified in the *hPolicy* parameter. The only method supported is binary version 0x020A.

1. unsigned long RRPC\_FWDeleteMainModeRule(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in, string, ref] LPCWSTR pRuleId
5. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. This handle MUST be of the [FW\_STORE\_TYPE\_DYNAMIC](#Section_37ebed958abf472c8b4b7a510a2a6baa) store.

**pRuleId:** This parameter is the pointer to a STRING that is the ID of the main mode rule the client deletes from the specified store.

This ID can be obtained by enumerating main mode rules using the RRPC\_FWEnumMainModeRules(Opnum 36) where the ID is returned in the [FW\_MM\_RULE](#Section_4b3fc163fede434c90cb557992caa5da) structure.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specified set referenced by the **wszRuleID** member string of the FW\_MM\_RULE data type is not found in the policy store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method either is incorrect or is required and not specified. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWDeleteAllMainModeRules (Opnum 35)

The RRPC\_FWDeleteAllMainModeRules (Opnum 35) method requests the server to delete all the main mode rules in the policy contained in the policy store referenced by the handle specified in the *hPolicy* parameter. The only method supported is binary version 0x020A.

1. unsigned long RRPC\_FWDeleteAllMainModeRules(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy
4. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. This handle MUST be of the [FW\_STORE\_TYPE\_DYNAMIC](#Section_37ebed958abf472c8b4b7a510a2a6baa) store.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicy* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWEnumMainModeRules (Opnum 36)

The RRPC\_FWEnumMainModeRules (Opnum 36) method requests the server to return all the main mode rules contained in the store referenced by the *hPolicy* handle. The method returns a linked list of all the main mode rule objects. The only method supported is binary version 0x020A.

1. unsigned long RRPC\_FWEnumMainModeRules(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in] unsigned long dwFilteredByStatus,
5. [in] unsigned long dwProfileFilter,
6. [in] unsigned short wFlags,
7. [out, ref] unsigned long\* pdwNumRules,
8. [out] PFW\_MM\_RULE\* ppMMRules
9. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. This handle MUST be of the [FW\_STORE\_TYPE\_DYNAMIC](#Section_37ebed958abf472c8b4b7a510a2a6baa) store.

**dwFilteredByStatus:** This parameter is a combination of flags from the [FW\_RULE\_STATUS\_CLASS](#Section_81877aa634c74664adf4ff8d5e0c0bc6) enumeration. This method uses this bitmask to determine which rules will be returned. Rules that contain a status code of the class specified by this parameter will be returned in the linked list.

**dwProfileFilter:** This parameter is a combination of flags from the [FW\_PROFILE\_TYPE](#Section_7704e238174d4a5eb8095f3787dd8acc) enumeration. This method also uses this parameter to determine which rules will be returned. Rules that contain a profile specified by this parameter will be returned in the linked list.

**wFlags:** This parameter is a combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumRules:** This is an output parameter that on success MUST be equal to the number of rules returned.

**ppMMRules:** This is an output parameter that on success contains a linked list of [FW\_MM\_RULE](#Section_4b3fc163fede434c90cb557992caa5da) data types.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *dwProfileFilter* parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWQueryFirewallRules (Opnum 37)

The RRPC\_FWQueryFirewallRules (Opnum 37) method requests the server to return all the firewall rules that match the specified query object that are contained in the store referenced by the *hPolicy* handle. The method returns a linked list of all the firewall rule objects. The only method supported is binary version 0x020A.

1. unsigned long RRPC\_FWQueryFirewallRules(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in] PFW\_QUERY pQuery,
5. [in] unsigned short wFlags,
6. [out, ref] unsigned long\* pdwNumRules,
7. [out] PFW\_RULE2\_10\* ppRule
8. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. This handle MUST be of the [FW\_STORE\_TYPE\_DYNAMIC](#Section_37ebed958abf472c8b4b7a510a2a6baa) store.

**pQuery:** This parameter represents the query object that the client uses to specify which main mode rules MUST be retrieved from the store. The query object MUST be valid, as specified in the definition of the [FW\_QUERY](#Section_51b69145317a4368a6e57159e946ff14) data type.

**wFlags:** This parameter is a combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumRules:** This is an output parameter that on success MUST be equal to the number of rules returned.

**ppRule:** This is an output parameter that on success contains a linked list of [FW\_RULE2\_10](#Section_4a9b9f60968948a981541ceb6d20edeb) data types.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *pQuery* parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWQueryConnectionSecurityRules (Opnum 38)

The RRPC\_FWQueryConnectionSecurityRules (Opnum 38) method requests the server to return all the connection security rules that match the specified query object that are contained in the store referenced by the *hPolicy* handle. The method returns a linked list of all the connection security rule objects. The only method supported is binary version 0x020A.

1. unsigned long RRPC\_FWQueryConnectionSecurityRules(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in] PFW\_QUERY pQuery,
5. [in] unsigned short wFlags,
6. [out, ref] unsigned long\* pdwNumRules,
7. [out] PFW\_CS\_RULE2\_10\* ppRules
8. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. This handle MUST be of the [FW\_STORE\_TYPE\_DYNAMIC](#Section_37ebed958abf472c8b4b7a510a2a6baa) store.

**pQuery:** This parameter represents the query object that the client uses to specify which main mode rules MUST be retrieved from the store. The query object MUST be valid, as specified in the definition of the [FW\_QUERY](#Section_51b69145317a4368a6e57159e946ff14) data type.

**wFlags:** This parameter is a combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08), which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumRules:** This is an output parameter that on success MUST be equal to the number of rules returned.

**ppRules:** This is an output parameter that on success contains a linked list of [FW\_CS\_RULE2\_10](#Section_8f1281f0bb53491fa552742a1a922137) data types.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *pQuery* parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWQueryMainModeRules (Opnum 39)

The RRPC\_FWQueryMainModeRules (Opnum 39) method requests the server to return all the main mode rules that match the specified query object that are contained in the store referenced by the *hPolicy* handle. The method returns a linked list of all the main mode rule objects. The only method supported is binary version 0x020A.

1. unsigned long RRPC\_FWQueryMainModeRules(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in] PFW\_QUERY pQuery,
5. [in] unsigned short wFlags,
6. [out, ref] unsigned long\* pdwNumRules,
7. [out] PFW\_MM\_RULE ppMMRules
8. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. This handle MUST be of the [FW\_STORE\_TYPE\_DYNAMIC](#Section_37ebed958abf472c8b4b7a510a2a6baa) store.

**pQuery:** This parameter represents the query object that the client uses to specify which main mode rules MUST be retrieved from the store. The query object MUST be valid, as specified in the definition of the [FW\_QUERY](#Section_51b69145317a4368a6e57159e946ff14) data type.

**wFlags:** This parameter is a combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumRules:** This is an output parameter that on success MUST be equal to the number of rules returned.

**ppMMRules:** This is an output parameter that on success contains a linked list of [FW\_MM\_RULE](#Section_4b3fc163fede434c90cb557992caa5da) data types.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *pQuery* parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWQueryAuthenticationSets (Opnum 40)

The RRPC\_FWQueryAuthenticationSets (Opnum 40) method requests the server to return all the authentication sets that match the specified query object that are contained in the store referenced by the *hPolicy* handle. The method returns a linked list of all the authentication set objects. The only method supported is binary version 0x020A.

1. unsigned long RRPC\_FWQueryAuthenticationSets(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)]
5. FW\_IPSEC\_PHASE IPsecPhase,
6. [in] PFW\_QUERY pQuery,
7. [in] unsigned short wFlags,
8. [out, ref] unsigned long\* pdwNumSets,
9. [out] PFW\_AUTH\_SET2\_10\* ppAuthSets
10. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. This handle MUST be of the [FW\_STORE\_TYPE\_DYNAMIC](#Section_37ebed958abf472c8b4b7a510a2a6baa) store.

**IPsecPhase:** This parameter specifies the specific [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) negotiation phase to which this set applies.

**pQuery:** This parameter represents the query object that the client wants to use to specify which main mode rules MUST be retrieved from the store. The query object MUST be valid, as specified in the definition of the [FW\_QUERY](#Section_51b69145317a4368a6e57159e946ff14) data type.

**wFlags:** This parameter is a combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumSets:** This is an output parameter that, on success, MUST be equal to the number of sets returned.

**ppAuthSets:** This is an output parameter that on success contains a linked list of [FW\_AUTH\_SET2\_10](#Section_c1acee0a747a4482b1590c96ca57fe29) data types.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *dwProfileFilter* parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWQueryCryptoSets (Opnum 41)

The RRPC\_FWQueryCryptoSets (Opnum 41) method requests the server to return all the crypto sets that match the specified query object that are contained in the store referenced by the *hPolicy* handle. The method returns a linked list of all the crypto set objects. The only method supported is binary version 0x020A.

1. unsigned long RRPC\_FWQueryCryptoSets(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)]
5. FW\_IPSEC\_PHASE IPsecPhase,
6. [in] PFW\_QUERY pQuery,
7. [in] unsigned short wFlags,
8. [out, ref] unsigned long\* pdwNumSets,
9. [out] PFW\_CRYPTO\_SET\* ppCryptoSets
10. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. This handle MUST be of the [FW\_STORE\_TYPE\_DYNAMIC](#Section_37ebed958abf472c8b4b7a510a2a6baa) store.

**IPsecPhase:** This parameter specifies the specific [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) negotiation phase to which this set applies.

**pQuery:** This parameter represents the query object that the client wants to use to specify which main mode rules MUST be retrieved from the store. The query object MUST be valid, as specified in the definition of the [FW\_QUERY](#Section_51b69145317a4368a6e57159e946ff14) data type.

**wFlags:** This parameter is a combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumSets:** This is an output parameter that, on success, MUST be equal to the number of sets returned.

**ppCryptoSets:** This is an output parameter that, on success, contains a linked list of [FW\_CRYPTO\_SET](#Section_a468fe9e113b4155a63d0db3aac12619) data types.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *pQuery* parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWEnumNetworks (Opnum 42)

The RRPC\_FWEnumNetworks (Opnum 42) method requests the server to return all the networks to which the host with the firewall and advanced security component is connected. The only method supported is binary version 0x020A.

1. unsigned long RRPC\_FWEnumNetworks(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [out] unsigned long pdwNumNetworks,
5. [out, size\_is(,\*pdwNumNetworks)]
6. PFW\_NETWORK\* ppNetworks
7. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. This handle MUST be of the [FW\_STORE\_TYPE\_DYNAMIC](#Section_37ebed958abf472c8b4b7a510a2a6baa) store.

**pdwNumNetworks:** This is an output parameter that, on success, MUST be equal to the number of networks returned.

**ppNetworks:** This is an output parameter that, on success, contains an array of [FW\_NETWORK](#Section_682c407a18024e9eb96a8c74315e5746) data types.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | A parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWEnumAdapters (Opnum 43)

The RRPC\_FWEnumAdapters (Opnum 43) method requests the server to return all the networks interfaces that the host with the firewall and advanced security component has. The only method supported is binary version 0x020A.

1. unsigned long RRPC\_FWEnumAdapters(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [out] unsigned long pdwNumAdapters,
5. [out, size\_is(,\*pdwNumAdapters)]
6. PFW\_ADAPTER\* ppAdapters
7. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. This handle MUST be of the [FW\_STORE\_TYPE\_DYNAMIC](#Section_37ebed958abf472c8b4b7a510a2a6baa) store.

**pdwNumAdapters:** This is an output parameter that, on success, MUST be equal to the number of networks returned.

**ppAdapters:** This is an output parameter that, on success, contains an array of [FW\_ADAPTER](#Section_24178f7ad5f749bda7ee69fe2b02ff4f) data types.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | A parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWGetGlobalConfig2\_10 (Opnum 44)

The RRPC\_FWGetGlobalConfig2\_10 (Opnum 44) method retrieves the value of a global policy configuration option. The client specifies to the server from which store this value MUST be retrieved and in which specific configuration option it is interested. The method is only supported for binary versions 0x020A and 0x0214.

1. unsigned long RRPC\_FWGetGlobalConfig2\_10(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] unsigned short BinaryVersion,
4. [in] FW\_STORE\_TYPE StoreType,
5. [in, range(FW\_GLOBAL\_CONFIG\_INVALID+1, FW\_GLOBAL\_CONFIG\_MAX-1)]
6. FW\_GLOBAL\_CONFIG configID,
7. [in] unsigned long dwFlags,
8. [in, out, unique, size\_is(cbData), length\_is(\*pcbTransmittedLen)]
9. BYTE\* pBuffer,
10. [in] unsigned long cbData,
11. [in, out] unsigned long\* pcbTransmittedLen,
12. [out] unsigned long\* pcbRequired,
13. [out] FW\_RULE\_ORIGIN\_TYPE\* pOrigin
14. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**BinaryVersion:** This parameter specifies the RPC interface binary version. This implies versions of the methods and versions of the structures.

**StoreType:** This parameter specifies the policy store from which the client retrieves the configuration option value.

**configID:** This parameter specifies the specific global policy configuration option that the client is interested in retrieving.

**dwFlags:** This parameter is a combination of flags from the [FW\_CONFIG\_FLAGS](#Section_eb0dce3b08d34c32b9fec8bdaf8e256a) enumeration, which modifies the behavior of this method, as specified in the definition of the enumeration.

**pBuffer:** This is an input/output parameter. This parameter is a pointer to the buffer that the client provides to contain the value of the profile configuration option that is being requested.

**cbData:** This parameter is the size of the buffer to which the *pBuffer* parameter points.

**pcbTransmittedLen:** This is a pointer to an input and output parameter that specifies the length of the transmitted data within the buffer.

**pcbRequired:** This is a pointer to an output parameter that specifies the required minimum buffer size, in octets, for the method to be able to return the configuration value. This output parameter is nonzero only if the buffer (pointed to by *pBuffer* and whose size is *cbData*) was not big enough to contain the value.

**pOrigin:** This field is the origin of the configuration option, as specified in the [FW\_RULE\_ORIGIN\_TYPE](#Section_9d295321d75c41c0ab0d7a78df40f77c) enumeration. On success, it MUST be filled.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specific configuration option is not found within the policy. This means that it is not configured. If the option is not configured in any other store, the firewall uses a default value. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store type does not support this method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method either is incorrect or is required and not specified. This error can be returned because:   * The specific configuration option is not meant to be available in the specified store. * The specified configuration option is not defined. * One of the required values is not specified. * The buffer is not big enough to hold the specific value. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWGetConfig2\_10 (Opnum 45)

The RRPC\_FWGetConfig2\_10 (Opnum 45) method retrieves the value of a profile configuration option. The client specifies to the server from which store and profile this value MUST be retrieved and in which specific configuration option it is interested. The method is only supported for binary versions 0x020A and 0x0214.

1. unsigned long RRPC\_FWGetConfig2\_10(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in, range(FW\_GLOBAL\_CONFIG\_INVALID+1, FW\_GLOBAL\_CONFIG\_MAX-1)]
5. FW\_GLOBAL\_CONFIG configID,
6. [in] FW\_PROFILE\_TYPE Profile,
7. [in] unsigned long dwFlags,
8. [in, out, unique, size\_is(cbData), length\_is(\*pcbTransmittedLen)]
9. BYTE\* pBuffer,
10. [in] unsigned long cbData,
11. [in, out] unsigned long\* pcbTransmittedLen,
12. [out] unsigned long\* pcbRequired,
13. [out] FW\_RULE\_ORIGIN\_TYPE\* pOrigin
14. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**configID:** This parameter specifies the specific global policy configuration option that the client is interested in retrieving.

**Profile:** This parameter specifies from which specific profile this value MUST be retrieved.

**dwFlags:** This parameter is a combination of flags from the [FW\_CONFIG\_FLAGS](#Section_eb0dce3b08d34c32b9fec8bdaf8e256a) enumeration, which modifies the behavior of this method, as specified in the definition of the enumeration.

**pBuffer:** This is an input/output parameter. This parameter is a pointer to the buffer that the client provides to contain the value of the profile configuration option being requested.

**cbData:** This parameter is the size of the buffer to which the *pBuffer* parameter points.

**pcbTransmittedLen:** This is a pointer to an input and output parameter that specifies the length of the transmitted data within the buffer.

**pcbRequired:** This is a pointer to an output parameter that specifies the required minimum buffer size, in octets, for the method to be able to return the configuration value. This output parameter is nonzero only if the buffer (pointed to by *pBuffer* and whose size is *cbData*) was not big enough to contain the value.

**pOrigin:** This field is the origin of the configuration option, as specified in the [FW\_RULE\_ORIGIN\_TYPE](#Section_9d295321d75c41c0ab0d7a78df40f77c) enumeration. On success, it MUST be filled.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specific configuration option is not found within the policy. This means that it is not configured. If the option is not configured in any other store, the firewall uses a default value. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The store type specified does not support this method. |
| 0x000000EA  ERROR\_MORE\_DATA | The buffer is not big enough to hold the configuration option value. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method either is incorrect or is required and not specified. This error can be returned because:   * The specific configuration option is not meant to be available in the specified store. * The specified configuration option is not defined. * One of the required values is not specified. * The buffer is not big enough to hold the specific value. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWAddFirewallRule2\_10 (Opnum 46)

The RRPC\_FWAddFirewallRule2\_10 (Opnum 46) method requests the server to add the specified firewall rule in the policy contained in the policy store referenced by the handle specified in the *hPolicyStore* parameter. The method is only supported for binary versions 0x020A and 0x0214.

1. unsigned long RRPC\_FWAddFirewallRule2\_10(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] PFW\_RULE2\_10 pRule,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pRule:** This parameter represents the firewall rule that the client wants to add to the store. The rule MUST be a valid rule, as specified in the definition of the [FW\_RULE2\_10](#Section_af151922b0924e5c9b694dca6b6d2ffc) data type.

**pStatus:** This output parameter is the status code of the rule as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out on return.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x000000B7  ERROR\_ALREADY\_EXISTS | The specified rule has a rule ID that already exists in the specified store. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method either is incorrect or is required and not specified. This error can be returned because:   * The *pRule* object did not pass the firewall rule validations specified in the definition of the [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b) data type. * One of the required values is not specified. * A policy store does not support rules with profile conditions other than ALL profiles. * The *wszLocalApplication* parameter contains a string that at enforcement time does not represent a valid file path.[<28>](#Appendix_A_28" \o "Product behavior note 28) |

**Exceptions Thrown:** No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method adds a firewall rule in the firewall linked list of the memory representation of the store being modified. It also writes through and saves the rule on disk. If called on an online store, the firewall rule is also enforced.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWSetFirewallRule2\_10 (Opnum 47)

The RRPC\_FWSetFirewallRule2\_10 (Opnum 47) method requests the server to modify the specified firewall rule in the policy contained in the policy store referenced by the handle specified in the *hPolicyStore* parameter. The method is only supported for binary versions 0x020A and 0x0214.

1. unsigned long RRPC\_FWSetFirewallRule2\_10(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] PFW\_RULE2\_10 pRule,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pRule:** This parameter represents the firewall rule that the client wants to add to the store. The rule MUST be a valid rule, as specified in the definition of the [FW\_RULE2\_10](#Section_af151922b0924e5c9b694dca6b6d2ffc) data type.

**pStatus:** This output parameter is the status code of the rule as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out on return.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specified rule referenced by the *wszRuleID* member string of the [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b) data type is not found in the policy store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method either is incorrect or is required and not specified. This error can be returned because:   * The *pRule* object did not pass the firewall rule validations specified in the definition of the FW\_RULE data type. * One of the required values is not specified. * A policy store does not support rules with profile conditions other than ALL profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWEnumFirewallRules2\_10 (Opnum 48)

The RRPC\_FWEnumFirewallRules2\_10 (Opnum 48) method requests the server to return all the firewall rules contained in the store referenced by the *hPolicyStore* handle. The method returns a linked list of all the firewall rule objects. The method is only supported for binary versions 0x020A and 0x0214.

1. unsigned long RRPC\_FWEnumFirewallRules2\_10(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] unsigned long dwFilteredByStatus,
5. [in] unsigned long dwProfileFilter,
6. [in] unsigned short wFlags,
7. [out, ref] unsigned long\* pdwNumRules,
8. [out] PFW\_RULE2\_10\* ppRules
9. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**dwFilteredByStatus:** This parameter is a combination of flags from the [FW\_RULE\_STATUS\_CLASS](#Section_81877aa634c74664adf4ff8d5e0c0bc6) enumeration. This method uses this bitmask to determine which rules will be returned. Rules that contain a status code of the specified class that match this parameter will be returned in the linked list.

**dwProfileFilter:** This parameter is a combination of flags from the [FW\_PROFILE\_TYPE](#Section_7704e238174d4a5eb8095f3787dd8acc) enumeration. This method also uses this parameter to determine which rules will be returned. Rules that contain a profile specified by this parameter will be returned in the linked list.

**wFlags:** This parameter is a combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumRules:** This is an output parameter that, on success, MUST be equal to the number of rules returned.

**ppRules:** This is an output parameter that, on success, contains a linked list of [FW\_RULE2\_10](#Section_af151922b0924e5c9b694dca6b6d2ffc) data types.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *dwProfileFilter* parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWAddConnectionSecurityRule2\_10 (Opnum 49)

The RRPC\_FWAddConnectionSecurityRule2\_10 (Opnum 49) method requests the server to add the specified connection security rule in the policy contained in the policy store referenced by the handle specified in the *hPolicyStore* parameter. The method is only supported for binary versions 0x020A and 0x0214.

1. unsigned long RRPC\_FWAddConnectionSecurityRule2\_10(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] PFW\_CS\_RULE2\_10 pRule,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pRule:** This parameter represents the firewall rule that the client adds to the store. The rule MUST be a valid rule, as specified in the definition of the [FW\_CS\_RULE2\_10](#Section_18277e4c628d44239fe3ba98601ba50d) data type.

**pStatus:** This output parameter is the status code of the rule as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out on return.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x000000B7  ERROR\_ALREADY\_EXISTS | The specified rule has a rule ID that already exists in the specified store. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method either is incorrect or is required and not specified. This error can be returned because:   * The *pRule* object did not pass the firewall rule validations specified in the definition of the [FW\_CS\_RULE](#Section_0d0641105f2e4b68aa63032c6cd5e4c6) data type. * One of the required values is not specified. * A policy store does not support rules with profile conditions other than ALL profiles. |

**Exceptions Thrown:** No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method adds a firewall rule in the firewall linked list of the memory representation of the store being modified. It also writes through and saves the rule on disk. If called on an online store, the firewall rule is also enforced.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWSetConnectionSecurityRule2\_10 (Opnum 50)

The RRPC\_FWSetConnectionSecurityRule2\_10 (Opnum 50) method requests the server to modify the specified connection security rule in the policy contained in the policy store referenced by the handle specified in the *hPolicyStore* parameter. The method is only supported for binary versions 0x020A and 0x0214.

1. unsigned long RRPC\_FWSetConnectionSecurityRule2\_10(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] PFW\_CS\_RULE2\_10 pRule,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pRule:** This parameter represents the connection security rule that the client wants to add to the store. The rule MUST be a valid rule, as specified in the definition of the [FW\_CS\_RULE2\_10](#Section_18277e4c628d44239fe3ba98601ba50d) data type.

**pStatus:** This output parameter is the status code of the rule as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out on return.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specified rule referenced by the **wszRuleID** member string of the [FW\_CS\_RULE](#Section_0d0641105f2e4b68aa63032c6cd5e4c6) data type is not found in the policy store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method either is incorrect or is required and not specified. This error can be returned because:   * The *pRule* object did not pass the firewall rule validations specified in the definition of the FW\_CS\_RULE data type. * One of the required values is not specified. * A policy store does not support rules with profile conditions other than ALL profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWEnumConnectionSecurityRules2\_10 (Opnum 51)

The RRPC\_FWEnumConnectionSecurityRules2\_10 (Opnum 51) method requests the server to return all the connection security rules contained in the store referenced by the *hPolicyStore* handle. The method returns a linked list of all the connection security rule objects. The method is only supported for binary versions 0x020A and 0x0214.

1. unsigned long RRPC\_FWEnumConnectionSecurityRules2\_10(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] unsigned long dwFilteredByStatus,
5. [in] unsigned long dwProfileFilter,
6. [in] unsigned short wFlags,
7. [out, ref] unsigned long\* pdwNumRules,
8. [out] PFW\_CS\_RULE2\_10\* ppRules
9. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**dwFilteredByStatus:** This parameter is a combination of flags from the [FW\_RULE\_STATUS\_CLASS](#Section_81877aa634c74664adf4ff8d5e0c0bc6) enumeration. This method uses this bitmask to determine which rules will be returned. Rules that contain a status code of the specified class that match this parameter will be returned in the linked list.

**dwProfileFilter:** This parameter is a combination of flags from the [FW\_PROFILE\_TYPE](#Section_7704e238174d4a5eb8095f3787dd8acc) enumeration. This method also uses this parameter to determine which rules will be returned. Rules that contain a profile specified by this parameter will be returned in the linked list.

**wFlags:** This parameter is a combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumRules:** This is an output parameter that on success MUST be equal to the number of rules returned.

**ppRules:** This is an output parameter that on success contains a linked list of [FW\_CS\_RULE2\_10](#Section_18277e4c628d44239fe3ba98601ba50d) data types.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *dwProfileFilter* parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWAddAuthenticationSet2\_10 (Opnum 52)

The RRPC\_FWAddAuthenticationSet2\_10 (Opnum 52) method requests the server to add the authentication set in the policy contained in the policy store referenced by the handle specified in the *hPolicy* parameter. The method is only supported for binary versions 0x020A and 0x0214.

1. unsigned long RRPC\_FWAddAuthenticationSet2\_10(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in] PFW\_AUTH\_SET2\_10 pAuth,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pAuth:** This parameter represents the authentication set that the client wants to add to the store. The set MUST be valid, as specified in the definition of the [FW\_AUTH\_SET2\_10](#Section_c1acee0a747a4482b1590c96ca57fe29) data type.

**pStatus:** This output parameter is the status code of the rule as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out on return.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x000000B7  ERROR\_ALREADY\_EXISTS | The specified rule has a rule ID that already exists in the specified store. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicy* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method either is incorrect or is required and not specified. This error can be returned because:   * The *pAuth* object did not pass the firewall rule validations specified in the definition of the [FW\_AUTH\_SET](#Section_bbec7ab70a1d49a49c8ea4784bbe6f16) data type. * One of the required values is not specified. |

**Exceptions Thrown:** No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method adds a firewall rule in the firewall linked list of the memory representation of the store being modified. It also writes through and saves the rule on disk. If the method is called on an online store, the firewall rule is also enforced.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWSetAuthenticationSet2\_10 (Opnum 53)

The RRPC\_FWSetAuthenticationSet2\_10 (Opnum 53) method requests the server to modify the specified authentication set in the policy contained in the policy store referenced by the handle specified in the *hPolicy* parameter. The method is only supported for binary versions 0x020A and 0x0214.

1. unsigned long RRPC\_FWSetAuthenticationSet2\_10(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in] PFW\_AUTH\_SET2\_10 pAuth,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pAuth:** This parameter represents the authentication set that the client wants to add to the store. The set MUST be valid, as specified in the definition of the [FW\_AUTH\_SET2\_10](#Section_c1acee0a747a4482b1590c96ca57fe29) data type.

**pStatus:** This output parameter is the status code of the rule as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out on return.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicy* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specified rule referenced by the **wszSetID** member string of the [FW\_AUTH\_SET](#Section_bbec7ab70a1d49a49c8ea4784bbe6f16) data type is not found in the policy store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method either is incorrect or is required and not specified. This error can be returned because:   * The *pAuth* object did not pass the firewall rule validations specified in the definition of the FW\_AUTH\_SET data type. * One of the required values is not specified. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWEnumAuthenticationSets2\_10 (Opnum 54)

The RRPC\_FWEnumAuthenticationSets2\_10 (Opnum 54) method requests the server to return all the authentication sets of the specified [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) phase contained in the store referenced by the *hPolicyStore* handle. The method returns a linked list of these objects. The method is only supported for binary versions 0x020A and 0x0214.

1. unsigned long RRPC\_FWEnumAuthenticationSets2\_10(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)]
5. FW\_IPSEC\_PHASE IpSecPhase,
6. [in] unsigned long dwFilteredByStatus,
7. [in] unsigned short wFlags,
8. [out, ref] unsigned long\* pdwNumAuthSets,
9. [out] PFW\_AUTH\_SET2\_10\* ppAuth
10. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**IpSecPhase:** This parameter specifies the specific IPsec negotiation phase to which this set applies.

**dwFilteredByStatus:** This parameter is a combination of flags from the [FW\_RULE\_STATUS\_CLASS](#Section_81877aa634c74664adf4ff8d5e0c0bc6) enumeration. This method uses this bitmask to determine which rules will be returned. Rules that contain a status code of the specified class that match this parameter will be returned in the linked list.

**wFlags:** This parameter is a combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumAuthSets:** This is an output parameter that on success MUST be equal to the number of sets returned.

**ppAuth:** This is an output parameter that, on success, contains a linked list of [FW\_AUTH\_SET2\_10](#Section_c1acee0a747a4482b1590c96ca57fe29) data types.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *dwProfileFilter* parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWAddCryptoSet2\_10 (Opnum 55)

The RRPC\_FWAddCryptoSet2\_10 (Opnum 55) method adds a cryptographic set in the cryptographic linked list of the memory representation of the store being modified. The method is only supported for binary versions 0x020A and 0x0214.

1. unsigned long RRPC\_FWAddCryptoSet2\_10(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in] PFW\_CRYPTO\_SET pCrypto,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pCrypto:** This parameter represents the cryptographic set that the client adds to the store. The set MUST be valid, as specified in the definition of the [FW\_CRYPTO\_SET](#Section_a468fe9e113b4155a63d0db3aac12619) data type.

**pStatus:** This output parameter is the status code of the rule as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out on return.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x000000B7  ERROR\_ALREADY\_EXISTS | The specified cryptographic set has a cryptographic set ID that already exists in the specified store. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicy* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method either is incorrect or is required and not specified. This error can be returned because:   * The *pCrypto* object did not pass the crypto set validations specified in the definition of the FW\_CRYPTO\_SET data type. * One of the required values is not specified. |

**Exceptions Thrown:** No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method adds a firewall rule in the firewall linked list of the memory representation of the store being modified. It also writes through and saves the rule on disk. If called on an online store, the firewall rule is also enforced.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWSetCryptoSet2\_10 (Opnum 56)

The RRPC\_FWSetCryptoSet2\_10 (Opnum 56) method requests the server to modify the specified cryptographic set in the policy contained in the policy store referenced by the handle specified in the *hPolicy* parameter. The method is only supported for binary versions 0x020A and 0x0214.

1. unsigned long RRPC\_FWSetCryptoSet2\_10(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in] PFW\_CRYPTO\_SET pCrypto,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pCrypto:** This parameter represents the cryptographic set that the client adds to the store. The set MUST be valid, as specified in the definition of the [FW\_CRYPTO\_SET](#Section_a468fe9e113b4155a63d0db3aac12619) data type.

**pStatus:** This output parameter is the status code of the rule as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out on return.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicy* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specified rule referenced by the **wszSetID** member string of the FW\_CRYPTO\_SET data type is not found in the policy store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method either is incorrect or is required and not specified. This error can be returned because:   * The *pCrypto* object did not pass the crypto set validations specified in the definition of the FW\_CRYPTO\_SET data type. * One of the required values is not specified. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWEnumCryptoSets2\_10 (Opnum 57)

The RRPC\_FWEnumCryptoSets2\_10 (Opnum 57) method requests the server to return all the cryptographic sets of the specified [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) phase contained in the store referenced by the *hPolicyStore* handle. The method returns a linked list of these objects. The method is only supported for binary versions 0x020A and 0x0214.

1. unsigned long RRPC\_FWEnumCryptoSets2\_10(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)]
5. FW\_IPSEC\_PHASE IpSecPhase,
6. [in] unsigned long dwFilteredByStatus,
7. [in] unsigned short wFlags,
8. [out, ref] unsigned long\* pdwNumSets,
9. [out] PFW\_CRYPTO\_SET\* ppCryptoSets
10. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**IpSecPhase:** This parameter specifies the specific IPsec negotiation phase to which this set applies.

**dwFilteredByStatus:** This parameter is a combination of flags from the [FW\_RULE\_STATUS\_CLASS](#Section_81877aa634c74664adf4ff8d5e0c0bc6) enumeration. This method uses this bitmask to determine which rules will be returned. Rules that contain a status code of the class specified that match this parameter will be returned in the linked list.

**wFlags:** This parameter is a combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumSets:** This is an output parameter that, on success, MUST be equal to the number of sets returned.

**ppCryptoSets:** This is an output parameter that, on success, contains a linked list of [FW\_CRYPTO\_SET](#Section_a468fe9e113b4155a63d0db3aac12619) data types.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *dwProfileFilter* parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWAddConnectionSecurityRule2\_20 (Opnum 58)

The RRPC\_FWAddConnectionSecurityRule2\_20 method requests the server to add the specified connection security rule in the policy contained in the policy store referenced by the handle specified in the *hPolicyStore* parameter. The method is only supported for binary version 0x0214.

1. unsigned unsigned long RRPC\_FWAddConnectionSecurityRule2\_20(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] PFW\_CS\_RULE pRule,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pRule:** This parameter represents the firewall rule that the client adds to the store. The rule MUST be a valid rule, as specified in the definition of the [FW\_CS\_RULE](#Section_0d0641105f2e4b68aa63032c6cd5e4c6) data type.

**pStatus:** This output parameter is the status code of the rule as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out on return.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x000000B7  ERROR\_ALREADY\_EXISTS | The specified rule has a rule ID that already exists in the specified store. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method either is incorrect or is required and not specified. This error can be returned because:   * The *pRule* object did not pass the firewall rule validations specified in the definition of the FW\_CS\_RULE data type. * One of the required values is not specified. * A policy store does not support rules with profile conditions other than ALL profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method adds a firewall rule in the firewall linked list of the memory representation of the store being modified. It also writes through and saves the rule on disk. If called on an online store, the firewall rule is also enforced.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWSetConnectionSecurityRule2\_20 (Opnum 59)

The RRPC\_FWSetConnectionSecurityRule2\_20 method requests the server to modify the specified connection security rule in the policy contained in the policy store referenced by the handle specified in the hPolicyStore parameter. The method is only supported for binary version 0x0214.

1. unsigned unsigned long RRPC\_FWModifyConnectionSecurityRule2\_20(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] PFW\_CS\_RULE pRule,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pRule:** This parameter represents the firewall rule that the client wants to add to the store. The rule MUST be a valid rule, as specified in the definition of the [FW\_CS\_RULE](#Section_0d0641105f2e4b68aa63032c6cd5e4c6) data type.

**pStatus:** This output parameter is the status code of the rule as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out on return.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specified rule referenced by the **wszRuleID** member string of the FW\_CS\_RULE data type is not found in the policy store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method either is incorrect or is required and not specified. This error can be returned because:   * The *pRule* object did not pass the firewall rule validations specified in the definition of the FW\_CS\_RULE data type. * One of the required values is not specified. * A policy store does not support rules with profile conditions other than ALL profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWEnumConnectionSecurityRules2\_20 (Opnum 60)

The RRPC\_FWEnumConnectionSecurityRules2\_20 (Opnum 60) method requests the server to return all the connection security rules contained in the store referenced by the *hPolicyStore* handle. The method returns a linked list of all the connection security rule objects. The method is only supported for binary version 0x0214.

1. unsigned long RRPC\_FWEnumConnectionSecurityRules2\_20(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] unsigned long dwFilteredByStatus,
5. [in] unsigned long dwProfileFilter,
6. [in] unsigned short wFlags,
7. [out, ref] unsigned long\* pdwNumRules,
8. [out] PFW\_CS\_RULE\* ppRules
9. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**dwFilteredByStatus:** This parameter is a combination of flags from the [FW\_RULE\_STATUS\_CLASS](#Section_81877aa634c74664adf4ff8d5e0c0bc6) enumeration. This method uses this bitmask to determine which rules will be returned. Rules that contain a status code of the specified class that match this parameter will be returned in the linked list.

**dwProfileFilter:** This parameter is a combination of flags from the [FW\_PROFILE\_TYPE](#Section_7704e238174d4a5eb8095f3787dd8acc) enumeration. This method also uses this parameter to determine which rules will be returned. Rules that contain a profile specified by this parameter will be returned in the linked list.

**wFlags:** This parameter is a combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumRules:** This is an output parameter that on success MUST be equal to the number of rules returned.

**ppRules:** This is an output parameter that on success contains a linked list of [FW\_CS\_RULE](#Section_0d0641105f2e4b68aa63032c6cd5e4c6) data types.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *dwProfileFilter* parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWQueryConnectionSecurityRules2\_20 (Opnum 61)

The RRPC\_FWQueryConnectionSecurityRules2\_20 (Opnum 61) method requests the server to return all the connection security rules that match the specified query object that are contained in the store referenced by the *hPolicy* handle. The method returns a linked list of all the connection security rule objects. The method is only supported for binary version 0x0214.

1. unsigned unsigned long RRPC\_FWQueryConnectionSecurityRules2\_20(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in] unsigned PFW\_QUERY pQuery,
5. [in] unsigned short wFlags,
6. [out, ref] unsigned long\* pdwNumRules,
7. [out] PFW\_CS\_RULE\* ppRules
8. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST be of the FW\_STORE\_TYPE\_DYNAMIC store.

**pQuery:** This parameter represents the query object that the client uses to specify which main mode rules MUST be retrieved from the store. The query object MUST be valid, as specified in the definition of the [FW\_QUERY](#Section_51b69145317a4368a6e57159e946ff14) data type.

**wFlags:** This parameter is a combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumRules:** This is an output parameter that on success MUST be equal to the number of rules returned.

**ppRules:** This is an output parameter that on success contains a linked list of [FW\_CS\_RULE](#Section_0d0641105f2e4b68aa63032c6cd5e4c6) data types.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *dwProfileFilter* parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWAddAuthenticationSet2\_20 (Opnum 62)

The RRPC\_FWAddAuthenticationSet2\_20 method requests the server to add the authentication set in the policy contained in the policy store referenced by the handle specified in the **hPolicy** parameter. The method is only supported for binary version 0x0214.

1. unsigned long RRPC\_FWAddAuthenticationSet2\_20(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in] PFW\_AUTH\_SET pAuth,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pAuth:** This parameter represents the authentication set the client wants to add to the store. The set MUST be valid, as specified in the definition of the [FW\_AUTH\_SET](#Section_bbec7ab70a1d49a49c8ea4784bbe6f16) data type.

**pStatus:** This output parameter is the status code of the rule as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out on return.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x000000B7  ERROR\_ALREADY\_EXISTS | The specified set has a set ID that already exists in the specified store. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicy* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect, or is required and not specified. This error can be returned because:   * The *pSet* object did not pass the firewall rule validations specified in the definition of the FW\_AUTH\_SET data type. * One of the required values is not specified. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15).

This method adds a firewall rule in the firewall linked list of the memory representation of the store being modified. It also writes through and saves the rule in disk. If the method is called on an online store, the firewall rule is also enforced.

The server MUST validate the client credentials to the administrator or network operator before executing this method.

#### RRPC\_FWSetAuthenticationSet2\_20 (Opnum 63)

The RRPC\_FWSetAuthenticationSet2\_20 method requests the server to modify the specified authentication set in the policy contained in the policy store referenced by the handle specified in the **hPolicy** parameter. The method is only supported for binary version 0x0214.

1. unsigned long RRPC\_FWSetAuthenticationSet2\_20(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in] PFW\_AUTH\_SET pAuth,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pAuth:** This parameter represents the authentication set that the client wants to add to the store. The set MUST be valid, as specified in the definition of the [FW\_AUTH\_SET](#Section_bbec7ab70a1d49a49c8ea4784bbe6f16) data type.

**pStatus:** This output parameter is the status code of the rule as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out on return.

**Return Values:** The method returns 0 if successful; if failed, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicy* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0X00000002  ERROR\_FILE\_NOT\_FOUND | The specified rule referenced by the **wszSetID** member string of the FW\_AUTH\_SET data type is not found in the policy store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect, or is required and not specified. This error can be returned because:   * The *pSet* object did not pass the firewall rule validations specified in the definition of the FW\_AUTH\_SET data type. * One of the required values is not specified. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15).

The server MUST validate the client credentials to the administrator or network operator before executing this method.

#### RRPC\_FWEnumAuthenticationSets2\_20 (Opnum 64)

The RRPC\_FWEnumAuthenticationSets2\_20 method requests the server to return all the authentication sets of the specified [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) phase contained in the store referenced in the **hPolicy** handle. The method returns a linked list of these objects. The method is only supported for binary version 0x0214.

1. unsigned long RRPC\_FWEnumAuthenticationSets2\_20(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)]
5. FW\_IPSEC\_PHASE IpSecPhase,
6. [in] DWORD dwFilteredByStatus,
7. [in] WORD wFlags,
8. [out] DWORD\* pdwNumAuthSets,
9. [out] PFW\_AUTH\_SET\* ppAuth
10. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**IpSecPhase:** This parameter specifies the specific IPsec negotiation phase to which this set applies.

**dwFilteredByStatus:** This parameter is a combination of flags from the [FW\_RULE\_STATUS\_CLASS](#Section_81877aa634c74664adf4ff8d5e0c0bc6) enumeration. This method uses this bitmask to determine which rules will be returned. Rules that contain a status code of the specified class that match this parameter will be returned in the linked list.

**wFlags:** This parameter is a combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumAuthSets:** This is an output parameter that, on success, MUST be equal to the number of sets returned.

**ppAuth:** This parameter represents the authentication set the client has added to the store. The set MUST be valid, as specified in the definition of the [FW\_AUTH\_SET](#Section_bbec7ab70a1d49a49c8ea4784bbe6f16) data type.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicy* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The dwProfileFilter parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15).

#### RRPC\_FWQueryAuthenticationSets2\_20 (Opnum 65)

The RRPC\_FWQueryAuthenticationSets2\_20 method requests the server to return all the authentication sets that match the specified query object that are contained in the store referenced in the **hPolicy** handle. The method returns a linked list of all the authentication set objects. The method is only supported for binary version 0x0214.

1. unsigned long RRPC\_FWQueryAuthenticationSets2\_20(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)]
5. FW\_IPSEC\_PHASE IpSecPhase,
6. [in] PFW\_QUERY pQuery,
7. [in] WORD wFlags,
8. [out, ref] DWORD\* pdwNumSets,
9. [out] PFW\_AUTH\_SET\* ppAuthSets
10. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST be of the FW\_STORE\_TYPE\_DYNAMIC store.

**IpSecPhase:** This parameter specifies the specific [**IPsec**](#gt_f8a5b7f0-25e0-4c81-9abf-00b56a580deb) negotiation phase to which this set applies.

**pQuery:** This parameter represents the query object that the client wants to use to specify which main mode rules MUST be retrieved from the store. The query object MUST be valid, as specified in the definition of the [FW\_QUERY](#Section_51b69145317a4368a6e57159e946ff14) data type.

**wFlags:** This parameter is a combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumSets:** This is an output parameter that, on success, MUST be equal to the number of sets returned.

**ppAuthSets:** This is an output parameter that, on success, contains a linked list of [FW\_AUTH\_SET](#Section_bbec7ab70a1d49a49c8ea4784bbe6f16) data types.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *dwProfileFilter* parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15).

The server MUST validate the client credentials to the administrator or network operator before executing this method.

#### RRPC\_FWAddFirewallRule2\_20 (Opnum 66)

The RRPC\_FWAddFirewallRule2\_20 method requests the server to add the specified firewall rule in the policy contained in the policy store referenced by the handle specified in the *hPolicyStore* parameter. The method is only supported for binary version 0x0214.

1. unsigned unsigned long RRPC\_FWAddFirewallRule2\_20(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] FW\_RULE2\_20 pRule,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pRule:** This parameter represents the firewall rule that the client adds to the store. The rule MUST be a valid rule, as specified in the definition of the [FW\_RULE2\_20](#Section_b166ba69c6d9488fa0813a4122002ced) data type.

**pStatus:** This output parameter is the status code of the rule as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out on return.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x000000B7  ERROR\_ALREADY\_EXISTS | The specified rule has a rule ID that already exists in the specified store. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method either is incorrect or is required and not specified. This error can be returned because:   * The *pRule* object did not pass the firewall rule validations specified in the definition of the [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b) data type. * One of the required values is not specified. * A policy store does not support rules with profile conditions other than ALL profiles. * The **wszLocalApplication** member contains a string that, at enforcement time, does not represent a valid file path. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method adds a firewall rule in the firewall linked list of the memory representation of the store being modified. It also writes through and saves the rule on disk. If called on an online store, the firewall rule is also enforced.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWSetFirewallRule2\_20 (Opnum 67)

The RRPC\_FWAddConnectionSecurityRule2\_20 method requests the server to modify the specified connection security rule in the policy contained in the policy store referenced by the handle specified in the *hPolicyStore* parameter. The method is only supported for binary version 0x0214.

1. unsigned unsigned long RRPC\_FWAddConnectionSecurityRule2\_20(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] PFW\_RULE2\_20 pRule,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**pRule:** This parameter represents the firewall rule that the client adds to the store. The rule MUST be a valid rule, as specified in the definition of the [FW\_RULE2\_20](#Section_b166ba69c6d9488fa0813a4122002ced) data type.

**pStatus:** This output parameter is the status code of the rule as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out on return.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specified rule referenced by the **wszRuleID** member string of the [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b) data type is not found in the policy store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method either is incorrect or is required and not specified. This error can be returned because:   * The *pRule* object did not pass the firewall rule validations specified in the definition of the FW\_RULE data type. * One of the required values is not specified. * A policy store does not support rules with profile conditions other than ALL profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWEnumFirewallRules2\_20 (Opnum 68)

The RRPC\_FWEnumFirewallRules2\_20 (Opnum 68) method requests the server to return all the firewall rules contained in the store referenced by the *hPolicyStore* handle. The method returns a linked list of all the firewall rule objects. The method is only supported for binary version 0x0214.

1. unsigned unsigned long RRPC\_FWEnumFirewallRules2\_20(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] unsigned long dwFilteredByStatus,
5. [in] unsigned long dwProfileFilter,
6. [in] unsigned short wFlags,
7. [out, ref] unsigned long\* pdwNumRules,
8. [out] PFW\_RULE2\_20\* ppRules
9. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST have read/write access rights.

**dwFilteredByStatus:** This parameter is a combination of flags from the [FW\_RULE\_STATUS\_CLASS](#Section_81877aa634c74664adf4ff8d5e0c0bc6) enumeration. This method uses this bitmask to determine which rules will be returned. Rules that contain a status code of the specified class that match this parameter will be returned in the linked list.

**dwProfileFilter:** This parameter is a combination of flags from the [FW\_PROFILE\_TYPE](#Section_7704e238174d4a5eb8095f3787dd8acc) enumeration. This method also uses this parameter to determine which rules will be returned. Rules that contain a profile specified by this parameter will be returned in the linked list.

**wFlags:** This parameter is a combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumRules:** This is an output parameter that on success MUST be equal to the number of rules returned.

**ppRules:** This is an output parameter that on success contains a linked list of [FW\_RULE2\_20](#Section_b166ba69c6d9488fa0813a4122002ced) data types.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. The error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *dwProfileFilter* parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWQueryFirewallRules2\_20 (Opnum 69)

The RRPC\_FWQueryFirewallRules2\_20 (Opnum 69) method requests the server to return all the firewall rules that match the specified query object that are contained in the store referenced by the *hPolicy* handle. The method returns a linked list of all the connection security rule objects. The method is only supported for binary version 0x0214.

1. unsigned unsigned long RRPC\_FWQueryFirewallRules2\_20(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicy,
4. [in] unsigned PFW\_QUERY pQuery,
5. [in] unsigned short wFlags,
6. [out, ref] unsigned long\* pdwNumRules,
7. [out] PFW\_RULE2\_20\* ppRules
8. );

**rpcConnHandle:** This parameter is an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicy:** This input parameter is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method. The handle MUST be of the FW\_STORE\_TYPE\_DYNAMIC store.

**pQuery:** This parameter represents the query object that the client uses to specify which main mode rules MUST be retrieved from the store. The query object MUST be valid, as specified in the definition of the [FW\_QUERY](#Section_51b69145317a4368a6e57159e946ff14) data type.

**wFlags:** This parameter is a combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumRules:** This is an output parameter that on success MUST be equal to the number of rules returned.

**ppRules:** This is an output parameter that on success contains a linked list of [FW\_RULE2\_20](#Section_b166ba69c6d9488fa0813a4122002ced) data types.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *pQuery* parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWAddFirewallRule2\_24 (Opnum 70)

The RRPC\_FWAddFirewallRule2\_24 method requests the server to add the specified firewall rule in the policy contained in the policy store that is referenced by the handle specified in the *hPolicyStore* parameter. The method is only supported for binary version 0x0218.

1. DWORD RRPC\_FWAddFirewallRule2\_24(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] PFW\_RULE2\_24 pRule,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** An input parameter that is an FW\_POLICY\_STORE\_HANDLE data type. This parameter MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method (section [3.1.4.1](#Section_2157E39A1BF04E0CABAE0811FD918B11)). The handle MUST have read/write access rights.

**pRule:** Represents the firewall rule that the client adds to the store. The rule MUST be a valid rule, as specified in the definition of the [FW\_RULE2\_24](#Section_4f5bc5fd4a3c46a598dc98b31b73aa89) data type.

**pStatus:** An output parameter that is the status code of the rule, as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out on return.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x000000B7  ERROR\_ALREADY\_EXISTS | The specified rule has a rule ID that already exists in the specified store. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. This error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect or is required but not specified. This error can be returned in the following cases:  - The *pRule* object did not pass the firewall rule validations specified in the definition of the [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b) data type.  - One of the required values is not specified.  - A policy store does not support rules with profile conditions other than ALL profiles.  - The **wszLocalApplication** member of the rule contains a string that, at enforcement time, does not represent a valid file path. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method adds a firewall rule in the firewall linked list of the memory representation of the store being modified. It also writes through and saves the rule on disk. If called on an online store, the firewall rule is also enforced.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_B0C9335283714D63ABCAB0CC8DBCC3D7)) before executing this method.

#### RRPC\_FWSetFirewallRule2\_24 (Opnum 71)

The RRPC\_FWSetFirewallRule2\_24 method requests the server to modify the specified connection security rule in the policy contained in the policy store that is referenced by the handle specified in the *hPolicyStore* parameter. The method is only supported for binary version 0x0218.

1. DWORD RRPC\_FWSetFirewallRule2\_24(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] PFW\_RULE2\_24 pRule,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** An input parameter that is an FW\_POLICY\_STORE\_HANDLE data type. This parameter MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method (section [3.1.4.1](#Section_2157E39A1BF04E0CABAE0811FD918B11)). The handle MUST have read/write access rights.

**pRule:** Represents the firewall rule that the client modifies in the store. The rule MUST be a valid rule, as specified in the definition of the [FW\_RULE2\_24](#Section_4f5bc5fd4a3c46a598dc98b31b73aa89) data type.

**pStatus:** An output parameter that is the status code of the rule, as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out on return.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. This error is also returned if the client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specified rule referenced by the **wszRuleID** member string of the [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b) data type is not found in the policy store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect or is required but not specified. This error can be returned in the following cases:  - The *pRule* object did not pass the firewall rule validations specified in the definition of the FW\_RULE data type.  - One of the required values is not specified.  - A policy store does not support rules with profile conditions other than ALL profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_B0C9335283714D63ABCAB0CC8DBCC3D7)) before executing this method.

#### RRPC\_FWEnumFirewallRules2\_24 (Opnum 72)

The RRPC\_FWEnumFirewallRules2\_24 method requests the server to return all the firewall rules contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all the firewall rule objects. The method is only supported for binary version 0x0218.

1. DWORD RRPC\_FWEnumFirewallRules2\_24(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] DWORD dwFilteredByStatus,
5. [in] DWORD dwProfileFilter,
6. [in] WORD wFlags,
7. [out, ref] DWORD\* pdwNumRules,
8. [out] PFW\_RULE2\_24\* ppRules
9. );

**rpcConnHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** An input parameter that is an **FW\_POLICY\_STORE\_HANDLE** data type. This parameter MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method (section [3.1.4.1](#Section_2157E39A1BF04E0CABAE0811FD918B11)). The handle MUST have read/write access rights.

**dwFilteredByStatus:** A combination of flags from the **FW\_RULE\_STATUS\_CLASS** enumeration. This method uses this bitmask to determine which rules will be returned. Rules that contain a status code of the specified class that match this parameter will be returned in the linked list.

**dwProfileFilter:** A combination of flags from the **FW\_PROFILE\_TYPE** enumeration. This method also uses this parameter to determine which rules will be returned. Rules that contain a profile specified by this parameter will be returned in the linked list.

**wFlags:** A combination of flags from the **FW\_ENUM\_RULES\_FLAGS** enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumRules:** An output parameter that, on success, MUST be equal to the number of rules returned.

**ppRules:** An output parameter that, on success, contains a linked list of **FW\_RULE2\_24** data types.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. This error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *dwProfileFilter* parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_B0C9335283714D63ABCAB0CC8DBCC3D7)) before executing this method.

#### RRPC\_FWQueryFirewallRules2\_24 (Opnum 73)

The RRPC\_FWQueryFirewallRules2\_24 method requests the server to return all the firewall rules that match the specified query object that are contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all the connection security rule objects. The method is only supported for binary version 0x0218.

1. DWORD RRPC\_FWQueryFirewallRules2\_24(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] PFW\_QUERY pQuery,
5. [in] WORD wFlags,
6. [out, ref] DWORD\* pdwNumRules,
7. [out] PFW\_RULE2\_24\* ppRules
8. );

**rpcConnHandle:** An [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore:** An input parameter that is an **FW\_POLICY\_STORE\_HANDLE** data type. The data type MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method (section [3.1.4.1](#Section_2157E39A1BF04E0CABAE0811FD918B11)). The handle MUST be of the **FW\_STORE\_TYPE\_DYNAMIC** store.

**pQuery:** Represents the query object that the client uses to specify which main mode rules MUST be retrieved from the store. The query object MUST be valid, as specified in the definition of the **FW\_QUERY** data type.

**wFlags:** A combination of flags from the **FW\_ENUM\_RULES\_FLAGS** enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumRules:** An output parameter that, on success, MUST be equal to the number of rules returned.

**ppRules:** An output parameter that, on success, contains a linked list of **FW\_RULE2\_24**\_data types.

**Return Values:** The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *pQuery* parameter contains invalid conditions. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_B0C9335283714D63ABCAB0CC8DBCC3D7)) before executing this method.

#### RRPC\_FWAddFirewallRule2\_25 (Opnum 74)

The RRPC\_FWAddFirewallRule2\_25 method requests the server to add the specified firewall rule in the policy contained in the policy store that is referenced by the handle specified in the *hPolicyStore* parameter. The method is only supported for binary version 0x0219.

1. DWORD RRPC\_FWAddFirewallRule2\_25(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] PFW\_RULE2\_25 pRule,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle**: An RPC binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore**: An input parameter that is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. This parameter MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method (section [3.1.4.1](#Section_2157E39A1BF04E0CABAE0811FD918B11)). The handle MUST have read/write access rights.

**pRule**: Represents the firewall rule that the client adds to the store. The rule MUST be a valid rule, as specified in the definition of the **FW\_RULE2\_25** data type (section [2.2.104](#Section_a744eb6c7e364b4599eea0ad078b31fd)).

**pStatus**: An output parameter that is the status code of the rule, as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out on return.

**Return Values**: The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x000000B7  ERROR\_ALREADY\_EXISTS | The specified rule has a rule ID that already exists in the specified store. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. This error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect or is required but not specified. This error can be returned in the following cases:  - The *pRule* object did not pass the firewall rule validations specified in the definition of the [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b) data type.  - One of the required values is not specified.  - A policy store does not support rules with profile conditions other than ALL profiles.  - The **wszLocalApplication** member of the rule contains a string that, at enforcement time, does not represent a valid file path. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method adds a firewall rule in the firewall linked list of the memory representation of the store being modified. It also writes through and saves the rule on disk. If called on an online store, the firewall rule is also enforced.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_B0C9335283714D63ABCAB0CC8DBCC3D7)) before executing this method.

#### RRPC\_FWSetFirewallRule2\_25 (Opnum 75)

The RRPC\_FWSetFirewallRule2\_25 method requests the server to modify the specified connection security rule in the policy contained in the policy store that is referenced by the handle specified in the *hPolicyStore* parameter. The method is only supported for binary version 0x0219.

1. DWORD RRPC\_FWSetFirewallRule2\_25(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] PFW\_RULE2\_25 pRule,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle**: An RPC binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore**: An input parameter that is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. This parameter MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method (section [3.1.4.1](#Section_2157E39A1BF04E0CABAE0811FD918B11)). The handle MUST have read/write access rights.

**pRule**: Represents the firewall rule that the client modifies in the store. The rule MUST be a valid rule, as specified in the definition of the **FW\_RULE2\_25** data type (section [2.2.104](#Section_a744eb6c7e364b4599eea0ad078b31fd)).

**pStatus**: An output parameter that is the status code of the rule, as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out on return.

**Return Values**: The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. This error is also returned if the client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specified rule referenced by the **wszRuleID** member string of the [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b) data type is not found in the policy store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect or is required but not specified. This error can be returned in the following cases:  - The *pRule* object did not pass the firewall rule validations specified in the definition of the FW\_RULE data type.  - One of the required values is not specified.  - A policy store does not support rules with profile conditions other than ALL profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_B0C9335283714D63ABCAB0CC8DBCC3D7)) before executing this method.

#### RRPC\_FWEnumFirewallRules2\_25 (Opnum 76)

The RRPC\_FWEnumFirewallRules2\_25 method requests the server to return all the firewall rules contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all the firewall rule objects. The method is only supported for binary version 0x0219.

1. DWORD RRPC\_FWEnumFirewallRules2\_25(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] DWORD dwFilteredByStatus,
5. [in] DWORD dwProfileFilter,
6. [in] WORD wFlags,
7. [out, ref] DWORD\* pdwNumRules,
8. [out] PFW\_RULE2\_25\* ppRules
9. );

**rpcConnHandle**: An RPC binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore**: An input parameter that is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. This parameter MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method (section [3.1.4.1](#Section_2157E39A1BF04E0CABAE0811FD918B11)). The handle MUST have read/write access rights.

**dwFilteredByStatus**: A combination of flags from the [FW\_RULE\_STATUS\_CLASS](#Section_81877aa634c74664adf4ff8d5e0c0bc6) enumeration. This method uses this bitmask to determine whether rules should be returned. Rules that contain a status code of the specified class that match this parameter will be returned in the linked list.

**dwProfileFilter**: A combination of flags from the [FW\_PROFILE\_TYPE](#Section_7704e238174d4a5eb8095f3787dd8acc) enumeration. This method also uses this parameter to determine whether rules should be returned. Rules that contain a profile specified by this parameter will be returned in the linked list.

**wFlags**: A combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumRules**: An output parameter that, on success, MUST be equal to the number of rules returned.

**ppRules**: An output parameter that, on success, contains a linked list of **FW\_RULE2\_25** data types (section [2.2.104](#Section_a744eb6c7e364b4599eea0ad078b31fd)).

**Return Values**: The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. This error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *dwProfileFilter* parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_B0C9335283714D63ABCAB0CC8DBCC3D7)) before executing this method.

#### RRPC\_FWQueryFirewallRules2\_25 (Opnum 77)

The RRPC\_FWQueryFirewallRules2\_25 method requests the server to return all the firewall rules that match the specified query object that are contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all the connection security rule objects. The method is only supported for binary version 0x0219.

1. DWORD RRPC\_FWQueryFirewallRules2\_25(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] PFW\_QUERY pQuery,
5. [in] WORD wFlags,
6. [out, ref] DWORD\* pdwNumRules,
7. [out] PFW\_RULE2\_25\* ppRules
8. );

**rpcConnHandle**: An RPC binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore**: An input parameter that is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. This parameter MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method (section [3.1.4.1](#Section_2157E39A1BF04E0CABAE0811FD918B11)). The handle MUST be of the FW\_STORE\_TYPE\_DYNAMIC store (section [2.2.1](#Section_37ebed958abf472c8b4b7a510a2a6baa)).

**pQuery**: Represents the query object that the client uses to specify which main mode rules MUST be retrieved from the store. The query object MUST be valid, as specified in the definition of the [FW\_QUERY](#Section_51b69145317a4368a6e57159e946ff14) data type.

**wFlags**: A combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumRules**: An output parameter that, on success, MUST be equal to the number of rules returned.

**ppRules**: An output parameter that, on success, contains a linked list of **FW\_RULE2\_25** data types (section [2.2.104](#Section_a744eb6c7e364b4599eea0ad078b31fd)).

**Return Values**: The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *pQuery* parameter contains invalid conditions. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_B0C9335283714D63ABCAB0CC8DBCC3D7)) before executing this method.

#### RRPC\_FWAddFirewallRule2\_26 (Opnum 78)

The RRPC\_FWAddFirewallRule2\_26 method requests the server to add the specified firewall rule in the policy contained in the policy store that is referenced by the handle specified in the *hPolicyStore* parameter. The method is only supported for binary version 0x021A.

1. DWORD RRPC\_FWAddFirewallRule2\_26(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] PFW\_RULE2\_26 pRule,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle**: An RPC binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore**: An input parameter that is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. This parameter MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method (section [3.1.4.1](#Section_2157E39A1BF04E0CABAE0811FD918B11)). The handle MUST have read/write access rights.

**pRule**: Represents the firewall rule that the client adds to the store. The rule MUST be a valid rule, as specified in the definition of the [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b)2\_26 data type.

**pStatus**: An output parameter that is the status code of the rule, as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out on return.

**Return Values**: The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x000000B7  ERROR\_ALREADY\_EXISTS | The specified rule has a rule ID that already exists in the specified store. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. This error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect or is required but not specified. This error can be returned in the following cases:  - The *pRule* object did not pass the firewall rule validations specified in the definition of the FW\_RULE data type.  - One of the required values is not specified.  - A policy store does not support rules with profile conditions other than ALL profiles.  - The **wszLocalApplication** member of the rule contains a string that, at enforcement time, does not represent a valid file path. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method adds a firewall rule in the firewall linked list of the memory representation of the store being modified. It also writes through and saves the rule on disk. If called on an online store, the firewall rule is also enforced.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_B0C9335283714D63ABCAB0CC8DBCC3D7)) before executing this method.

#### RRPC\_FWSetFirewallRule2\_26 (Opnum 79)

The RRPC\_FWSetFirewallRule2\_26 method requests the server to modify the specified connection security rule in the policy contained in the policy store that is referenced by the handle specified in the *hPolicyStore* parameter. The method is only supported for binary version 0x021A.

1. DWORD RRPC\_FWSetFirewallRule2\_26(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] PFW\_RULE2\_26 pRule,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle**: An RPC binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore**: An input parameter that is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. This parameter MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method (section [3.1.4.1](#Section_2157E39A1BF04E0CABAE0811FD918B11)). The handle MUST have read/write access rights.

**pRule**: Represents the firewall rule that the client modifies in the store. The rule MUST be a valid rule, as specified in the definition of the [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b)2\_26 data type.

**pStatus**: An output parameter that is the status code of the rule, as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration. This field is filled out on return.

**Return Values**: The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. This error is also returned if the client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specified rule referenced by the **wszRuleID** member string of the FW\_RULE data type is not found in the policy store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect or is required but not specified. This error can be returned in the following cases:  - The *pRule* object did not pass the firewall rule validations specified in the definition of the FW\_RULE data type.  - One of the required values is not specified.  - A policy store does not support rules with profile conditions other than ALL profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_B0C9335283714D63ABCAB0CC8DBCC3D7)) before executing this method.

#### RRPC\_FWEnumFirewallRules2\_26 (Opnum 80)

The RRPC\_FWEnumFirewallRules2\_26 method requests the server to return all the firewall rules contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all the firewall rule objects. The method is only supported for binary version 0x021A.

1. DWORD RRPC\_FWEnumFirewallRules2\_26(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] DWORD dwFilteredByStatus,
5. [in] DWORD dwProfileFilter,
6. [in] WORD wFlags,
7. [out, ref] DWORD\* pdwNumRules,
8. [out] PFW\_RULE2\_26\* ppRules
9. );

**rpcConnHandle**: An RPC binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore**: An input parameter that is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. This parameter MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method (section [3.1.4.1](#Section_2157E39A1BF04E0CABAE0811FD918B11)). The handle MUST have read/write access rights.

**dwFilteredByStatus**: A combination of flags from the [FW\_RULE\_STATUS\_CLASS](#Section_81877aa634c74664adf4ff8d5e0c0bc6) enumeration. This method uses this bitmask to determine whether rules should be returned. Rules that contain a status code of the specified class that match this parameter will be returned in the linked list.

**dwProfileFilter**: A combination of flags from the [FW\_PROFILE\_TYPE](#Section_7704e238174d4a5eb8095f3787dd8acc) enumeration. This method also uses this parameter to determine whether rules should be returned. Rules that contain a profile specified by this parameter will be returned in the linked list.

**wFlags**: A combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumRules**: An output parameter that, on success, MUST be equal to the number of rules returned.

**ppRules**: An output parameter that, on success, contains a linked list of [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b)2\_26 data types.

**Return Values**: The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. This error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *dwProfileFilter* parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_B0C9335283714D63ABCAB0CC8DBCC3D7)) before executing this method.

#### RRPC\_FWQueryFirewallRules2\_26 (Opnum 81)

The RRPC\_FWQueryFirewallRules2\_26 method requests the server to return all the firewall rules that match the specified query object that are contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all the connection security rule objects. The method is only supported for binary version 0x021A.

1. DWORD RRPC\_FWQueryFirewallRules2\_26(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] PFW\_QUERY pQuery,
5. [in] WORD wFlags,
6. [out, ref] DWORD\* pdwNumRules,
7. [out] PFW\_RULE2\_26\* ppRules
8. );

**rpcConnHandle**: An RPC binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore**: An input parameter that is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type. This parameter MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method (section [3.1.4.1](#Section_2157E39A1BF04E0CABAE0811FD918B11)). The handle MUST be of the FW\_STORE\_TYPE\_DYNAMIC store (section [2.2.1](#Section_37ebed958abf472c8b4b7a510a2a6baa)).

**pQuery**: Represents the query object that the client uses to specify which main mode rules MUST be retrieved from the store. The query object MUST be valid, as specified in the definition of the [FW\_QUERY](#Section_51b69145317a4368a6e57159e946ff14) data type.

**wFlags**: A combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration, which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumRules**: An output parameter that, on success, MUST be equal to the number of rules returned.

**ppRules**: An output parameter that, on success, contains a linked list of [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b)2\_26 data types.

**Return Values**: The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *pQuery* parameter contains invalid conditions. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_B0C9335283714D63ABCAB0CC8DBCC3D7)) before executing this method.

#### RRPC\_FWAddFirewallRule2\_27 (Opnum 82)

The RRPC\_FWAddFirewallRule2\_27 method requests the server to add the specified firewall rule to the policy contained in the policy store that is referenced by the handle specified in the *hPolicyStore* parameter. The method is only supported for binary version 0x021B.

1. DWORD RRPC\_FWAddFirewallRule2\_27(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] PFW\_RULE pRule,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle**: An RPC binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore**: An input parameter that is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type (section 2.2.93). This parameter MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method (section [3.1.4.1](#Section_2157e39a1bf04e0cabae0811fd918b11)). The handle MUST have read/write access rights.

**pRule**: Represents the firewall rule that the client adds to the store. The rule MUST be a valid rule, as specified in the definition of the [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b) data type (section 2.2.36).

**pStatus**: An output parameter that is the status code of the rule, as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration (section 2.2.23). This field is filled out on return.

**Return Values**: The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x000000B7  ERROR\_ALREADY\_EXISTS | The specified rule has a rule ID that already exists in the specified store. |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. This error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect or is required but not specified. This error can be returned in the following cases:  - The *pRule* object did not pass the firewall rule validations specified in the definition of the FW\_RULE data type (section 2.2.36).  - One of the required values is not specified.  - A policy store does not support rules with profile conditions other than ALL profiles.  - The **wszLocalApplication** member of the rule contains a string that, at enforcement time, does not represent a valid file path. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

This method adds a firewall rule in the firewall linked list of the memory representation of the store being modified. It also writes through and saves the rule on disk. If called on an online store, the firewall rule is also enforced.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWSetFirewallRule2\_27 (Opnum 83)

The RRPC\_FWSetFirewallRule2\_27 method requests the server to modify the specified connection security rule in the policy contained in the policy store that is referenced by the handle specified in the *hPolicyStore* parameter. The method is only supported for binary version 0x021B.

1. DWORD RRPC\_FWSetFirewallRule2\_27(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] PFW\_RULE pRule,
5. [out] FW\_RULE\_STATUS\* pStatus
6. );

**rpcConnHandle**: An RPC binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore**: An input parameter that is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type (section 2.2.93). This parameter MUST contain an opened policy store handle, successfully opened with the RRPC\_FWOpenPolicyStore (Opnum 0) method (section [3.1.4.1](#Section_2157e39a1bf04e0cabae0811fd918b11)). The handle MUST have read/write access rights.

**pRule**: Represents the firewall rule that the client modifies in the store. The rule MUST be a valid rule, as specified in the definition of the [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b) data type (section 2.2.36).

**pStatus**: An output parameter that is the status code of the rule, as specified by the [FW\_RULE\_STATUS](#Section_c4fd81e638dd47c58a0fa98a11cf190e) enumeration (section 2.2.23). This field is filled out on return.

**Return Values**: The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000032  ERROR\_NOT\_SUPPORTED | The specified store does not support this method; the store might be read-only. |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. This error is also returned if the client does not have the required credentials to call the method. |
| 0x00000002  ERROR\_FILE\_NOT\_FOUND | The specified rule referenced by the **wszRuleID** member string of the FW\_RULE data type (section 2.2.36) is not found in the policy store. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | One of the parameters of this method is incorrect or is required but not specified. This error can be returned in the following cases:  - The **pRule** object did not pass the firewall rule validations specified in the definition of the FW\_RULE data type (section 2.2.36).  - One of the required values is not specified.  - A policy store does not support rules with profile conditions other than ALL profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWEnumFirewallRules2\_27 (Opnum 84)

The **RRPC\_FWEnumFirewallRules2\_27** method requests the server to return all the firewall rules contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all the firewall rule objects. The method is only supported for binary version 0x021B.

1. DWORD RRPC\_FWEnumFirewallRules2\_27(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] DWORD dwFilteredByStatus,
5. [in] DWORD dwProfileFilter,
6. [in] WORD wFlags,
7. [out, ref] DWORD\* pdwNumRules,
8. [out] PFW\_RULE\* ppRules
9. );

**rpcConnHandle**: An RPC binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore**: An input parameter that is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type (section 2.2.93). This parameter MUST contain an opened policy store handle, successfully opened with the **RRPC\_FWOpenPolicyStore (Opnum 0)** method (section [3.1.4.1](#Section_2157e39a1bf04e0cabae0811fd918b11)). The handle MUST have read/write access rights.

**dwFilteredByStatus**: A combination of flags from the [FW\_RULE\_STATUS\_CLASS](#Section_81877aa634c74664adf4ff8d5e0c0bc6) enumeration (section 2.2.24). This method uses this bitmask to determine whether rules should be returned. Rules that contain a status code of the specified class that match this parameter will be returned in the linked list.

**dwProfileFilter**: A combination of flags from the [FW\_PROFILE\_TYPE](#Section_7704e238174d4a5eb8095f3787dd8acc) enumeration (section 2.2.2). This method also uses this parameter to determine whether rules should be returned. Rules that contain a profile specified by this parameter will be returned in the linked list.

**wFlags**: A combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration (section 2.2.32), which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumRules**: An output parameter that, on success, MUST be equal to the number of rules returned.

**ppRules**: An output parameter that, on success, contains a linked list of [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b) data types (section 2.2.36).

**Return Values**: The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The *hPolicyStore* handle was not opened with read/write access rights. This error is also returned if the client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *dwProfileFilter* parameter contains invalid profiles. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

#### RRPC\_FWQueryFirewallRules2\_27 (Opnum 85)

The **RRPC\_FWQueryFirewallRules2\_27** method requests the server to return all the firewall rules that match the specified query object that are contained in the store that is referenced by the *hPolicyStore* handle. The method returns a linked list of all the connection security rule objects. The method is only supported for binary version 0x021B.

1. DWORD RRPC\_FWQueryFirewallRules2\_27(
2. [in] FW\_CONN\_HANDLE rpcConnHandle,
3. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
4. [in] PFW\_QUERY pQuery,
5. [in] WORD wFlags,
6. [out, ref] DWORD\* pdwNumRules,
7. [out] PFW\_RULE ppRules
8. );

**rpcConnHandle**: An RPC binding handle that connects to the RPC interface of the Firewall and Advanced Security Protocol.

**hPolicyStore**: An input parameter that is an [FW\_POLICY\_STORE\_HANDLE](#Section_88f6ea4aa9464747bb93f142f1760752) data type (section 2.2.93). This parameter MUST contain an opened policy store handle, successfully opened with the **RRPC\_FWOpenPolicyStore (Opnum 0)** method (section [3.1.4.1](#Section_2157e39a1bf04e0cabae0811fd918b11)). The handle MUST be the FW\_STORE\_TYPE\_DYNAMIC policy store type (section [2.2.1](#Section_37ebed958abf472c8b4b7a510a2a6baa)).

**pQuery**: Represents the query object that the client uses to specify which main mode rules MUST be retrieved from the store. The query object MUST be valid, as specified in the definition of the [FW\_QUERY](#Section_51b69145317a4368a6e57159e946ff14) data type (section 2.2.92).

**wFlags**: A combination of flags from the [FW\_ENUM\_RULES\_FLAGS](#Section_aff3633c43b24cefba63f35b204a8a08) enumeration (section 2.2.32), which modifies the behavior of the method and performs operations on the rules before returning them in the linked list.

**pdwNumRules**: An output parameter that, on success, MUST be equal to the number of rules returned.

**ppRules**: An output parameter that, on success, contains a linked list of [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b) data types (section 2.2.36).

**Return Values**: The method returns 0 if successful; if it fails, it returns a nonzero error code. The field can take any specific error code value, as specified in [[MS-ERREF]](%5bMS-ERREF%5d.pdf#Section_1bc92ddfb79e413cbbaa99a5281a6c90). The following return values are common.

| Return value/code | Description |
| --- | --- |
| 0x00000005  ERROR\_ACCESS\_DENIED | The client does not have the required credentials to call the method. |
| 0x00000057  ERROR\_INVALID\_PARAMETER | The *pQuery* parameter contains invalid conditions. |

**Exceptions Thrown**: No exceptions are thrown beyond those thrown by the underlying RPC protocol, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15). If any lower-layer errors are reported by RPC exception, this exception is converted to an error code and reported to higher-layer protocols via the return value.

The server MUST validate that the client is authorized to perform the requested operation (as defined in section [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7)) before executing this method.

### Timer Events

No timer events are required on the server other than the events maintained in the underlying [**RPC transport**](#gt_c2eeb200-3cd0-4916-966e-d7d6bff1737a).

### Other Local Events

The following sections describe the abstract interfaces available.

#### AddPortInUse

AddPortInUse is an abstract interface called by applications and services on the local computer to add a **PortInUse** object to the server's **PortsInUse** collection. The interface is defined as follows:

1. void AddPortInUse([in] PortInUse portToAdd);

**Input Parameter**: portToAdd: The **PortInUse** object to be added.

**Output Parameter**: None.

#### DeletePortInUse

DeletePortInUse is an abstract interface called by applications and services on the local computer to delete a **PortInUse** object from the server's **PortsInUse** collection. The interface is defined as follows:

1. void DeletePortInUse([in] PortInUse portToDelete);

**Input Parameter**: portToDelete: The **PortInUse** object to be deleted. If the port is not found in the server's **PortsInUse** collection, the method has no effect.

**Output Parameter**: None.

#### AddDefaultFirewallRule

AddDefaultFirewallRule is an abstract interface called by applications and services on the local computer to add a new **FirewallRule** object to the **FirewallRules** collection in the server's **DefaultsStore**. The interface is defined as follows:

1. void AddDefaultFirewallRule([in] FirewallRule ruleToAdd);

**Input Parameter**: ruleToAdd: The **FirewallRule** object to be added.

**Output Parameter**: None.

#### SetGroupPolicyRSoPStore

SetGroupPolicyRSoPStore is an abstract interface used to set the state of the **GroupPolicyRSoPStore** object. This interface is typically invoked by an implementation of [[MS-GPFAS]](%5bMS-GPFAS%5d.pdf#Section_46e8d583a4ce4c43b399566afb1eec7f) in order to notify the server of a policy change. See [MS-GPFAS] section 3.2.5 for details.

The server MUST replace the contents of **GroupPolicyRSoPStore** with the contents of the newSettings object. The server then MUST merge the existing contents of **LocalStore** with the new contents of **GroupPolicyRSoPStore** (as described in section [3.1.1](#Section_43507d538955416db913dfb27dc76b17)) and store the result in **DynamicStore**. The server MUST invoke the abstract interface [SetEffectiveFirewallPolicy (section 3.1.6.6)](#Section_0b159c9216fa42a69a33b68f0e040a98) with the contents of **DynamicStore**. The interface is defined as follows:

1. void SetGroupPolicyRSoPStore(
2. [in] PolicyStore newSettings
3. );

**Input Parameter**: newSettings: A **PolicyStore** object containing the new settings for the **GroupPolicyRSoPStore**.

**Output Parameter**: None.

#### IsComputerInCommonCriteriaMode

IsComputerInCommonCriteriaMode is an abstract interface exposed by the host operating system and invoked by the MS-FASP server to determine whether the local computer is conforming to all the security functional requirements specified in [[CCITSE3.1-3]](https://go.microsoft.com/fwlink/?LinkId=211804), Part 2. The algorithm for computing the return value is implementation-specific.[<29>](#Appendix_A_29" \o "Product behavior note 29) The interface is defined as follows:

1. bool IsComputerInCommonCriteriaMode();

**Input Parameter**: None.

**Output Parameter**: None.

#### SetEffectiveFirewallPolicy

SetEffectiveFirewallPolicy is an abstract interface exposed by the host operating system and invoked by the MS-FASP server whenever the effective firewall policy changes. The algorithm for processing the new policy settings is implementation-specific.[<30>](#Appendix_A_30" \o "Product behavior note 30) The interface is defined as follows:

1. void SetEffectiveFirewallPolicy(
2. [in] PolicyStore newEffectivePolicy
3. );

**Input Parameter**: newEffectivePolicy: A **PolicyStore** object containing the new effective firewall policy for the local computer.

**Output Parameter**: None.

#### AddTrustTuple

AddTrustTuple is an abstract interface called by applications and services on the local computer to add a **TrustTuple** object to the server's **TrustTuples** collection. The interface is defined as follows:

1. void AddTrustTuple([in] TrustTuple tupleToAdd);

**Input Parameter**: tupleToAdd: The **TrustTuple** object to be added.

**Output Parameter**: None.

#### DeleteTrustTuple

DeleteTrustTuple is an abstract interface called by applications and services on the local computer to delete a **TrustTuple** object from the server's **TrustTuples** collection. The interface is defined as follows:

1. void DeleteTrustTuple([in] TrustTuple tupleToDelete);

**Input Parameter**: tupleToDelete: The **TrustTuple** object to be deleted. If the trust tuple is not found in the server's **TrustTuples** collection, the method has no effect.

**Output Parameter**: None.

## Client Details

### Abstract Data Model

None.

### Timers

No protocol timers are required other than those internal ones used in the [**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 creates an [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) association (or binding) to the server RPC before an RPC method is called. The client can create a separate association for each method invocation, or it can reuse an association for multiple invocations.

### Message Processing Events and Sequencing Rules

This protocol MUST indicate to the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) runtime that it is to perform a strict [**NDR**](#gt_9ebf9540-2c31-43bc-bc56-4a932faabf2d) data consistency check at target level 6.0, as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15) section 3.

The client SHOULD ignore errors returned from the RPC server and notify the application invoker of the error received. Otherwise, no special message processing is required on the client beyond the processing required in the underlying RPC protocol.

### Timer Events

No protocol timer events are required on the client other than those internal ones maintained in the underlying [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331), as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15).

### Other Local Events

No local events are required on the client other than those internal ones maintained in the underlying [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331), as specified in [[MS-RPCE]](%5bMS-RPCE%5d.pdf#Section_290c38b192fe422991e64fc376610c15).

# Protocol Examples

## Opening a Policy Store

Before a client application can perform most of the operations, it opens a policy store handle. The protocol sequence that opens a policy store is as follows.

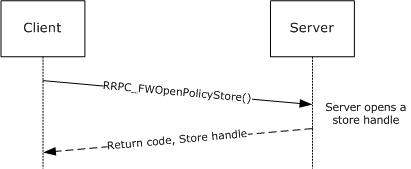


Figure 2: Opening a policy store

To open a policy store, the client first gets an rpcBinding to this interface in the server. Then the client simply calls the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) method to open the required store. In this case, the client chooses the local store.

1. FW\_POLICY\_STORE\_HANDLE hStore = NULL;
2. DWORD
3. RRPC\_FWOpenPolicyStore(
4. [in] FW\_CONN\_HANDLE rpcConnHandle = rpcBinding,
5. [in] WORD BinaryVersion = 0x0200,
6. [in] FW\_STORE\_TYPE StoreType = FW\_STORE\_TYPE\_LOCAL,
7. [in] FW\_POLICY\_ACCESS\_RIGHT AccessRight = FW\_POLICY\_ACCESS\_RIGHT\_READ\_WRITE,
8. [in] DWORD dwFlags = 0,
9. [out] PFW\_POLICY\_STORE\_HANDLE phPolicyStore = &hStore
10. );

## Adding a Firewall Rule

Once the client has a handle to an open policy store, the client can perform operations on the policy store. The protocol sequence that adds a firewall rule to the policy store is as follows.

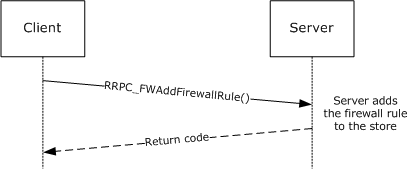


Figure 3: Adding a firewall rule

To add a firewall rule, the client application first fills an [FW\_RULE](#Section_8c008258166d46d49090f2ffaa01be4b) structure. The following examples fill this structure to represent a rule to allow inbound traffic to port 80 for the "c:\servers\MyWebServer.exe" application, which is also a service with the WebServerSVC name. The example also places this rule in the "HTTPWebServer" rule group.

1. FW\_PORT\_RANGE Port = {80,80};
2. FW\_RULE HTTPRule =
3. {
4. struct \_tag\_FW\_RULE \*pNext = NULL;
5. WORD wSchemaVersion = 0x0200;
6. WCHAR\* wszRuleId = L"{d439709f-d8ec-4d2e-b615-4cfcd9bacc05}";
7. WCHAR\* wszName = L"Web server requests";
8. WCHAR\* wszDescription = L"This rule allows incoming HTTP server requests";
9. DWORD dwProfiles = FW\_PROFILE\_TYPE\_ALL;
10. FW\_DIRECTION Direction = FW\_DIR\_IN;
11. WORD wIpProtocol = 0x0006;
12. FW\_PORTS LocalPorts = {0x0000, {1, &Port}};
13. FW\_PORTS RemotePorts = {0};
14. FW\_ADDRESSES LocalAddresses = {0};
15. FW\_ADDRESSES RemoteAddresses = {0};
16. FW\_INTERFACE\_LUIDS LocalInterfaceIds = {0};
17. DWORD dwLocalInterfaceTypes = 0;
18. WCHAR\* wszLocalApplication = L"c:\servers\MyWebServer.exe";
19. WCHAR\* wszLocalService = L"WebServerSVC";
20. FW\_RULE\_ACTION Action = FW\_RULE\_ACTION\_ALLOW;
21. WORD wFlags = FW\_RULE\_FLAGS\_ACTIVE;
22. WCHAR\* wszRemoteMachineAuthorizationList = NULL;
23. WCHAR\* wszRemoteUserAuthorizationList = NULL;
24. WCHAR\* wszEmbeddedContext = L"HTTP WebServer";
25. FW\_OS\_PLATFORM\_LIST PlatformValidityList = {0};
27. FW\_RULE\_STATUS Status = FW\_RULE\_STATUS\_OK;
28. FW\_RULE\_ORIGIN\_TYPE Origin = 0;
29. WCHAR\* wszGPOName =NULL;
30. DWORD Reserved = 0;
31. PFW\_OBJECT\_METADATA pMetaData = NULL;
32. };

Once the FW\_RULE structure is filled out, the client can simply invoke the [**RPC**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) [RRPC\_FWAddFirewallRule](#Section_4d7baada78fd469a9c292885e5d3b5d0) method, passing the required parameters as follows.

1. DWORD
2. RRPC\_FWAddFirewallRule(
3. [in] FW\_CONN\_HANDLE rpcConnHandle = rpcBinding,
4. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore = hStore,
5. [in] PFW\_RULE pRule = &HTTPRule
6. );

If the return code is FW\_ERROR\_ALREADY\_EXISTS, the rule exists in the store. The client can try using a different Rule ID or bubble up the error.

## Enumerating the Firewall Rules

To enumerate the firewall rules that the server is enforcing in the store, the client calls the RRPC\_FWEnumFirewallRules (Opnum 9) method. The protocol sequence that enumerates firewall rules from the policy store is as follows:

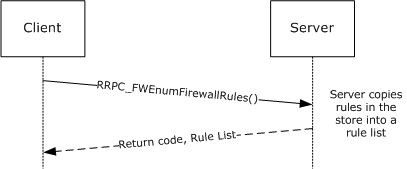


Figure 4: Enumerating firewall rules

In this case example, the client enumerates rules in the current profile and filters by FW\_RULE\_STATUS\_CLASS\_OK and FW\_RULE\_STATUS\_CLASS\_PARTIALLY\_IGNORED.

1. PFW\_RULE pRules = NULL;
2. DWORD dwNumRules = 0;
3. DWORD
4. RRPC\_FWEnumFirewallRules(
5. [in] FW\_CONN\_HANDLE rpcConnHandle = rpcBinding ,
6. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore = hStore,
7. [in] DWORD dwFilteredByStatus =
8. FW\_RULE\_STATUS\_CLASS\_OK | FW\_RULE\_STATUS\_CLASS\_PARTIALLY\_IGNORED,
9. [in] DWORD dwProfileFilter = FW\_PROFILE\_TYPE\_CURRENT,
10. [in] WORD wFlag = 0
11. [out, ref] DWORD \*pdwNumRules = &dwNumRules,
12. [out] PFW\_RULE \*ppRules = &pRules
13. );

## Closing a Policy Store Handle

Once a client application has finished managing the policy, it closes the policy store handle. The protocol sequence that closes a policy store follows.

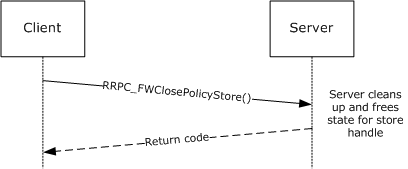


Figure 5: Closing a policy store

To close the handle, the client simply passes the handle to the close method.

1. DWORD
2. RRPC\_FWClosePolicyStore(
3. [in] FW\_CONN\_HANDLE rpcConnHandle = rpcBinding,
4. [in, out] PFW\_POLICY\_STORE\_HANDLE phPolicyStore = &hStore
5. );

# Security

## Security Considerations for Implementers

The enumeration methods require the server to return the correct number of objects linked in the returned linked list. For example, the **DWORD** variable passed in the *pdwNumRules* parameter of RRPC\_FWEnumFirewallRules (Opnum 9) must be equal to the actual number of rules returned in *ppRules*.

However, the client cannot assume that the server is accurate in the actual object count. The client can allocate a buffer based on the rule count; however, while filling the buffer, the client has to actively validate that the number of objects in the buffer does not exceed the object count. Failure to do this validation could result in buffer overruns on the client.

## Index of Security Parameters

| Security Parameter | Section |
| --- | --- |
| [**Remote procedure call (RPC)**](#gt_8a7f6700-8311-45bc-af10-82e10accd331) authentication | [2.1](#Section_81eb95d6df6349b6905265c99664e71f) |
| The required permissions to call each of the methods of the protocol interface | [3.1.4](#Section_b0c9335283714d63abcab0cc8dbcc3d7) |

# Appendix A: Full IDL

For ease of implementation, the full [**IDL**](#gt_73177eec-4092-420f-92c5-60b2478df824) is provided below.

1. import "ms-dtyp.idl";
2. cpp\_quote("#ifndef \_\_FIREWALL\_H\_")
3. cpp\_quote("#define FW\_CURRENT\_BINARY\_VERSION (FW\_VERSION(2,27))")
4. cpp\_quote("#define FW\_CURRENT\_SCHEMA\_VERSION (FW\_VERSION(2,27))")
5. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
6. \* \*
7. \* Firewall Policy Stores structures \*
8. \* \*
9. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
10. typedef enum \_tag\_FW\_STORE\_TYPE
11. {
12. FW\_STORE\_TYPE\_INVALID,
13. FW\_STORE\_TYPE\_GP\_RSOP, //read-only
14. FW\_STORE\_TYPE\_LOCAL,
15. FW\_STORE\_TYPE\_NOT\_USED\_VALUE\_3, //read-only
16. FW\_STORE\_TYPE\_NOT\_USED\_VALUE\_4,
17. FW\_STORE\_TYPE\_DYNAMIC,
18. FW\_STORE\_TYPE\_GPO,
19. FW\_STORE\_TYPE\_DEFAULTS,
20. FW\_STORE\_TYPE\_NOT\_USED\_VALUE\_8,
21. FW\_STORE\_TYPE\_NOT\_USED\_VALUE\_9,
22. FW\_STORE\_TYPE\_NOT\_USED\_VALUE\_10,
23. FW\_STORE\_TYPE\_NOT\_USED\_VALUE\_11,
24. FW\_STORE\_TYPE\_MAX,
25. } FW\_STORE\_TYPE;
26. typedef enum \_tag\_FW\_TRANSACTIONAL\_STATE
27. {
28. FW\_TRANSACTIONAL\_STATE\_NONE = 0,
29. FW\_TRANSACTIONAL\_STATE\_NO\_FLUSH,
30. FW\_TRANSACTIONAL\_STATE\_MAX
31. } FW\_TRANSACTIONAL\_STATE;
32. typedef
33. [v1\_enum]
34. enum \_tag\_FW\_PROFILE\_TYPE
35. {
36. FW\_PROFILE\_TYPE\_INVALID = 0,
37. FW\_PROFILE\_TYPE\_DOMAIN = 0x001,
38. FW\_PROFILE\_TYPE\_STANDARD = 0x002,
39. FW\_PROFILE\_TYPE\_PRIVATE = FW\_PROFILE\_TYPE\_STANDARD,
40. FW\_PROFILE\_TYPE\_PUBLIC = 0x004,
41. FW\_PROFILE\_TYPE\_ALL = 0x7FFFFFFF,
42. FW\_PROFILE\_TYPE\_CURRENT = 0x80000000,
43. FW\_PROFILE\_TYPE\_NONE = FW\_PROFILE\_TYPE\_CURRENT + 1
44. } FW\_PROFILE\_TYPE;
45. typedef enum \_tag\_FW\_POLICY\_ACCESS\_RIGHT
46. {
47. FW\_POLICY\_ACCESS\_RIGHT\_INVALID,
48. FW\_POLICY\_ACCESS\_RIGHT\_READ,
49. FW\_POLICY\_ACCESS\_RIGHT\_READ\_WRITE,
50. FW\_POLICY\_ACCESS\_RIGHT\_MAX
51. }FW\_POLICY\_ACCESS\_RIGHT;
52. typedef enum \_tag\_FW\_POLICY\_STORE\_FLAGS
53. {
54. FW\_POLICY\_STORE\_FLAGS\_NONE = 0x0000,
55. FW\_POLICY\_STORE\_FLAGS\_DELETE\_DYNAMIC\_RULES\_AFTER\_CLOSE = 0x0001,
56. FW\_POLICY\_STORE\_FLAGS\_OPEN\_GP\_CACHE = 0x0002,
57. FW\_POLICY\_STORE\_FLAGS\_USE\_GP\_CACHE = 0x0004,
58. FW\_POLICY\_STORE\_FLAGS\_SAVE\_GP\_CACHE = 0x0008,
59. FW\_POLICY\_STORE\_FLAGS\_NOT\_USED\_VALUE\_16 = 0x0010,
60. FW\_POLICY\_STORE\_FLAGS\_MAX = 0x0020
61. }FW\_POLICY\_STORE\_FLAGS;
62. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
63. \* \*
64. \* Firewall Rules structures \*
65. \* \*
66. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
67. typedef struct \_tag\_FW\_IPV4\_SUBNET
68. {
69. DWORD dwAddress;
70. DWORD dwSubNetMask;
71. } FW\_IPV4\_SUBNET, \*PFW\_IPV4\_SUBNET;
72. typedef struct \_tag\_FW\_IPV4\_SUBNET\_LIST
73. {
74. [range(0, 10000)]
75. DWORD dwNumEntries;
76. [size\_is(dwNumEntries)]
77. PFW\_IPV4\_SUBNET pSubNets;
78. } FW\_IPV4\_SUBNET\_LIST, \*PFW\_IPV4\_SUBNET\_LIST;
79. typedef struct \_tag\_FW\_IPV6\_SUBNET
80. {
81. BYTE Address[16];
82. [range(0, 128)]
83. DWORD dwNumPrefixBits;
84. } FW\_IPV6\_SUBNET, \*PFW\_IPV6\_SUBNET;
85. typedef struct \_tag\_FW\_IPV6\_SUBNET\_LIST
86. {
87. [range(0, 10000)]
88. DWORD dwNumEntries;
89. [size\_is(dwNumEntries)]
90. PFW\_IPV6\_SUBNET pSubNets;
91. } FW\_IPV6\_SUBNET\_LIST, \*PFW\_IPV6\_SUBNET\_LIST;
92. typedef struct \_tag\_FW\_IPV4\_ADDRESS\_RANGE
93. {
94. DWORD dwBegin;
95. DWORD dwEnd;
96. } FW\_IPV4\_ADDRESS\_RANGE, \*PFW\_IPV4\_ADDRESS\_RANGE;
97. typedef struct \_tag\_FW\_IPV6\_ADDRESS\_RANGE
98. {
99. BYTE Begin[16];
100. BYTE End[16];
101. } FW\_IPV6\_ADDRESS\_RANGE, \*PFW\_IPV6\_ADDRESS\_RANGE;
102. typedef struct \_tag\_FW\_IPV4\_RANGE\_LIST
103. {
104. [range(0, 10000)]
105. DWORD dwNumEntries;
106. [size\_is(dwNumEntries)]
107. PFW\_IPV4\_ADDRESS\_RANGE pRanges;
108. } FW\_IPV4\_RANGE\_LIST, \*PFW\_IPV4\_RANGE\_LIST;
109. typedef struct \_tag\_FW\_IPV6\_RANGE\_LIST
110. {
111. [range(0, 10000)]
112. DWORD dwNumEntries;
113. [size\_is(dwNumEntries)]
114. PFW\_IPV6\_ADDRESS\_RANGE pRanges;
115. } FW\_IPV6\_RANGE\_LIST, \*PFW\_IPV6\_RANGE\_LIST;
116. typedef struct \_tag\_FW\_PORT\_RANGE
117. {
118. WORD wBegin;
119. WORD wEnd;
120. } FW\_PORT\_RANGE, \*PFW\_PORT\_RANGE;
121. typedef struct \_tag\_FW\_PORT\_RANGE\_LIST
122. {
123. [range(0, 10000)]
124. DWORD dwNumEntries;
125. [size\_is(dwNumEntries)]
126. PFW\_PORT\_RANGE pPorts;
127. } FW\_PORT\_RANGE\_LIST, \*PFW\_PORT\_RANGE\_LIST;
128. typedef enum \_tag\_FW\_PORT\_KEYWORD
129. {
130. FW\_PORT\_KEYWORD\_NONE = 0x00,
131. FW\_PORT\_KEYWORD\_DYNAMIC\_RPC\_PORTS = 0x01,
132. FW\_PORT\_KEYWORD\_RPC\_EP = 0x02,
133. FW\_PORT\_KEYWORD\_TEREDO\_PORT = 0x04,
134. FW\_PORT\_KEYWORD\_IP\_TLS\_IN = 0x08,
135. FW\_PORT\_KEYWORD\_IP\_TLS\_OUT = 0x10,
136. FW\_PORT\_KEYWORD\_DHCP = 0x20,
137. FW\_PORT\_KEYWORD\_PLAYTO\_DISCOVERY = 0x40,
138. FW\_PORT\_KEYWORD\_MDNS = 0x80,
139. FW\_PORT\_KEYWORD\_CORTANA\_OUT = 0x100,
140. FW\_PORT\_KEYWORD\_MAX = 0x200,
141. FW\_PORT\_KEYWORD\_MAX\_V2\_1 = 0x08,
142. FW\_PORT\_KEYWORD\_MAX\_V2\_10 = 0x20,
143. FW\_PORT\_KEYWORD\_MAX\_V2\_20 = 0x80,
144. FW\_PORT\_KEYWORD\_MAX\_V2\_25 = 0x200
145. }FW\_PORT\_KEYWORD;
146. typedef struct \_tag\_FW\_PORTS
147. {
148. WORD wPortKeywords; // Bit-flags from FW\_PORT\_KEYWORD
149. FW\_PORT\_RANGE\_LIST Ports;
150. }FW\_PORTS,\*PFW\_PORTS;
151. cpp\_quote("#define FW\_ICMP\_CODE\_ANY (256)")
152. cpp\_quote("#define FW\_IP\_PROTOCOL\_ANY (256)")
153. typedef struct \_tag\_FW\_ICMP\_TYPE\_CODE
154. {
155. BYTE bType;
156. [range(0, 256)]
157. WORD wCode;
158. } FW\_ICMP\_TYPE\_CODE, \*PFW\_ICMP\_TYPE\_CODE;
159. typedef struct \_tag\_FW\_ICMP\_TYPE\_CODE\_LIST
160. {
161. [range(0, 10000)]
162. DWORD dwNumEntries;
163. [size\_is(dwNumEntries)]
164. PFW\_ICMP\_TYPE\_CODE pEntries;
165. } FW\_ICMP\_TYPE\_CODE\_LIST, \*PFW\_ICMP\_TYPE\_CODE\_LIST;
166. typedef struct \_tag\_FW\_INTERFACE\_LUIDS
167. {
168. [range(0, 10000)]
169. DWORD dwNumLUIDs;
170. [size\_is(dwNumLUIDs)]
171. GUID\* pLUIDs;
172. } FW\_INTERFACE\_LUIDS, \*PFW\_INTERFACE\_LUIDS;
173. typedef enum \_tag\_FW\_DIRECTION
174. {
175. FW\_DIR\_INVALID = 0,
176. FW\_DIR\_IN,
177. FW\_DIR\_OUT,
178. FW\_DIR\_MAX
179. } FW\_DIRECTION;
180. // Interface Types bitmap.
181. typedef enum \_tag\_FW\_INTERFACE\_TYPE
182. {
183. FW\_INTERFACE\_TYPE\_ALL = 0x0000,
184. FW\_INTERFACE\_TYPE\_LAN = 0x0001,
185. FW\_INTERFACE\_TYPE\_WIRELESS = 0x0002,
186. FW\_INTERFACE\_TYPE\_REMOTE\_ACCESS = 0x0004,
187. FW\_INTERFACE\_TYPE\_MOBILE\_BBAND = 0x0008,
188. FW\_INTERFACE\_TYPE\_MAX = 0x0010,
189. FW\_INTERFACE\_TYPE\_MAX\_V2\_23 = 0x0008,
190. } FW\_INTERFACE\_TYPE;
191. typedef enum \_tag\_FW\_ADDRESS\_KEYWORD
192. {
193. FW\_ADDRESS\_KEYWORD\_NONE = 0x0000,
194. FW\_ADDRESS\_KEYWORD\_LOCAL\_SUBNET = 0x0001,
195. FW\_ADDRESS\_KEYWORD\_DNS = 0x0002,
196. FW\_ADDRESS\_KEYWORD\_DHCP = 0x0004,
197. FW\_ADDRESS\_KEYWORD\_WINS = 0x0008,
198. FW\_ADDRESS\_KEYWORD\_DEFAULT\_GATEWAY = 0x0010,
199. FW\_ADDRESS\_KEYWORD\_INTRANET = 0x0020,
200. FW\_ADDRESS\_KEYWORD\_INTERNET = 0x0040,
201. FW\_ADDRESS\_KEYWORD\_PLAYTO\_RENDERERS= 0x0080,
202. FW\_ADDRESS\_KEYWORD\_REMOTE\_INTRANET = 0x0100,
203. FW\_ADDRESS\_KEYWORD\_MAX = 0x0200,
204. FW\_ADDRESS\_KEYWORD\_MAX\_V2\_10 = 0x0020
205. }FW\_ADDRESS\_KEYWORD;
206. typedef struct \_tag\_FW\_ADDRESSES
207. {
208. DWORD dwV4AddressKeywords; // Bit flags from FW\_ADDRESS\_KEYWORD
209. DWORD dwV6AddressKeywords; // Bit flags from FW\_ADDRESS\_KEYWORD
210. FW\_IPV4\_SUBNET\_LIST V4SubNets;
211. FW\_IPV4\_RANGE\_LIST V4Ranges;
212. FW\_IPV6\_SUBNET\_LIST V6SubNets;
213. FW\_IPV6\_RANGE\_LIST V6Ranges;
214. }FW\_ADDRESSES, \*PFW\_ADDRESSES;
215. typedef enum \_tag\_FW\_TRUST\_TUPLE\_KEYWORD
216. {
217. FW\_TRUST\_TUPLE\_KEYWORD\_NONE = 0x0000,
218. FW\_TRUST\_TUPLE\_KEYWORD\_PROXIMITY = 0x0001,
219. FW\_TRUST\_TUPLE\_KEYWORD\_PROXIMITY\_SHARING = 0x0002,
220. FW\_TRUST\_TUPLE\_KEYWORD\_WFD\_PRINT = 0x0004,
221. FW\_TRUST\_TUPLE\_KEYWORD\_WFD\_DISPLAY = 0x0008,
222. FW\_TRUST\_TUPLE\_KEYWORD\_WFD\_DEVICES = 0x0010,
223. FW\_TRUST\_TUPLE\_KEYWORD\_WFD\_KM\_DRIVER = 0x0020,
224. FW\_TRUST\_TUPLE\_KEYWORD\_UPNP = 0x0040,
225. FW\_TRUST\_TUPLE\_KEYWORD\_MAX = 0x0080,
226. FW\_TRUST\_TUPLE\_KEYWORD\_MAX\_V2\_20 = 0x0004,
227. FW\_TRUST\_TUPLE\_KEYWORD\_MAX\_V2\_26 = 0x0020,
228. }FW\_TRUST\_TUPLE\_KEYWORD;
229. typedef
230. [v1\_enum]
231. enum \_tag\_FW\_RULE\_STATUS
232. {
233. FW\_RULE\_STATUS\_OK = 0x00010000,
234. // The rule was parsed successfully from the store.
235. FW\_RULE\_STATUS\_PARTIALLY\_IGNORED = 0x00020000,
236. // The rule is from a later version of the service. Some fields
237. // were not understood and have been ignored. This may cause the
238. // rule to be less restrictive than on the version where it was
239. // created. To mitigate any risk from this fallback behavior,
240. // ensure that the original rule is as specific as possible. To
241. // avoid this fallback behavior, create version-specific GPO's, or
242. // apply a Platform condition to the rule.
243. FW\_RULE\_STATUS\_IGNORED = 0x00040000,
244. // The rule is from a newer schema version than the service, and
245. // the unknown fields could not be ignored. The whole rule was
246. // ignored.
247. FW\_RULE\_STATUS\_PARSING\_ERROR = 0x00080000,
248. // The service was unable to parse the rule.
249. FW\_RULE\_STATUS\_PARSING\_ERROR\_NAME = 0x00080001,
250. // The name contains invalid characters, or is an invalid length.
251. FW\_RULE\_STATUS\_PARSING\_ERROR\_DESC = 0x00080002,
252. // The description contains invalid characters, or is an invalid
253. // length.
254. FW\_RULE\_STATUS\_PARSING\_ERROR\_APP = 0x00080003,
255. // The application contains invalid characters, or is an invalid
256. // length.
257. FW\_RULE\_STATUS\_PARSING\_ERROR\_SVC = 0x00080004,
258. // The service contains invalid characters, or is an invalid length.
259. FW\_RULE\_STATUS\_PARSING\_ERROR\_RMA = 0x00080005,
260. // The authorized remote machines list contains invalid characters,
261. // or is an invalid length.
262. FW\_RULE\_STATUS\_PARSING\_ERROR\_RUA = 0x00080006,
263. // The authorized remote users list contains invalid characters, or
264. // is an invalid length.
265. FW\_RULE\_STATUS\_PARSING\_ERROR\_EMBD = 0x00080007,
266. // The group (sometimes called the embedded context) contains
267. // invalid characters, or is an invalid length.
268. FW\_RULE\_STATUS\_PARSING\_ERROR\_RULE\_ID = 0x00080008,
269. // The rule ID contains invalid characters, or is an invalid length.
270. FW\_RULE\_STATUS\_PARSING\_ERROR\_PHASE1\_AUTH = 0x00080009,
271. // The phase 1 auth set ID contains invalid characters, or is an
272. // invalid length.
273. FW\_RULE\_STATUS\_PARSING\_ERROR\_PHASE2\_CRYPTO = 0x0008000A,
274. // The quick mode crypto set ID contains invalid characters, or is
275. // an invalid length.
276. FW\_RULE\_STATUS\_PARSING\_ERROR\_PHASE2\_AUTH = 0x0008000B,
277. // The main mode crypto set ID contains invalid characters, or is
278. // an invalid length.
279. FW\_RULE\_STATUS\_PARSING\_ERROR\_RESOLVE\_APP = 0x0008000C,
280. // The application name could not be resolved.
281. FW\_RULE\_STATUS\_PARSING\_ERROR\_MAINMODE\_ID = 0x0008000D,
282. // This error value is not used.
283. FW\_RULE\_STATUS\_PARSING\_ERROR\_PHASE1\_CRYPTO = 0x0008000E,
284. // The phase 2 auth set ID contains invalid characters, or is an
285. // invalid length.
286. FW\_RULE\_STATUS\_PARSING\_ERROR\_REMOTE\_ENDPOINTS = 0x0008000F,
287. // The remote endpoints are invalid.
288. FW\_RULE\_STATUS\_PARSING\_ERROR\_REMOTE\_ENDPOINT\_FQDN = 0x00080010,
289. // The remote endpoint FQDN is invalid.
290. FW\_RULE\_STATUS\_PARSING\_ERROR\_KEY\_MODULE = 0x00080011,
291. // The choice of key modules is invalid.
292. FW\_RULE\_STATUS\_PARSING\_ERROR\_LUA = 0x00080012,
293. // The local user authorization list contains invalid characters,
294. // or is an invalid length.
295. FW\_RULE\_STATUS\_PARSING\_ERROR\_FWD\_LIFETIME = 0x00080013,
296. // The forward path SA lifetime is invalid.
297. FW\_RULE\_STATUS\_PARSING\_ERROR\_TRANSPORT\_MACHINE\_AUTHZ\_SDDL = 0x00080014,
298. // The transport rule machine SDDL is not valid.
299. FW\_RULE\_STATUS\_PARSING\_ERROR\_TRANSPORT\_USER\_AUTHZ\_SDDL = 0x00080015,
300. // The transport rule user SDDL is not valid.
301. FW\_RULE\_STATUS\_PARSING\_ERROR\_NETNAMES\_STRING = 0x00080016,
302. // A string of the network name structure is invalid.
303. FW\_RULE\_STATUS\_PARSING\_ERROR\_SECURITY\_REALM\_ID\_STRING = 0x00080017,
304. // A string for the security realm Id is invalid.
305. FW\_RULE\_STATUS\_PARSING\_ERROR\_FQBN\_STRING = 0x00080018,
306. // A string for the FQBN is invalid.
307. FW\_RULE\_STATUS\_SEMANTIC\_ERROR = 0x00100000,
308. // The rule was parsed successfully, but there was an unknown
309. // semantic error when processing the rule.
310. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_RULE\_ID = 0x00100010,
311. // The Rule ID was not specified.
312. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PORTS = 0x00100020,
313. // Mismatch in number of ports and ports buffer.
314. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PORT\_KEYW = 0x00100021,
315. // One of the port keywords is invalid.
316. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PORT\_RANGE = 0x00100022,
317. // An invalid port range was specified, or 0 was used as a port
318. // number.
319. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PORTRANGE\_RESTRICTION = 0x00100023,
320. // Port ranges are only allowed in connection security rules when
321. // the action is Do Not Secure.
322. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_V4\_SUBNETS = 0x00100040,
323. // Mismatch in number of V4 address subnets and subnets buffer.
324. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_V6\_SUBNETS = 0x00100041,
325. // Mismatch in number of V6 address subnets and subnets buffer.
326. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_V4\_RANGES = 0x00100042,
327. // Mismatch in number of V4 address ranges and ranges buffer.
328. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_V6\_RANGES = 0x00100043,
329. // Mismatch in number of V6 address ranges and ranges buffer.
330. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_RANGE = 0x00100044,
331. // The address range is invalid. The end address is less than the
332. // beginning address.
333. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_MASK = 0x00100045,
334. // One or more of the subnet masks is invalid.
335. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_PREFIX = 0x00100046,
336. // One or more of the address prefixes is invalid.
337. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_KEYW = 0x00100047,
338. // One or more of the address keywords are invalid.
339. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LADDR\_PROP = 0x00100048,
340. // Some of the keywords specified on the local address are only
341. // valid on the remote address.
342. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_RADDR\_PROP = 0x00100049,
343. // Some of the keywords specified on the remote address are only
344. // valid on the local address.
345. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_V6 = 0x0010004A,
346. // An unspecified, multicast, broadcast, or loopback IPv6 address
347. // was specified.
348. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LADDR\_INTF = 0x0010004B,
349. // A local address cannot be used in conjunction with an interface
350. // or interface type condition.
351. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_V4 = 0x0010004C,
352. // An unspecified, multicast, broadcast, or loopback IPv4 address
353. // was specified.
354. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TUNNEL\_ENDPOINT\_ADDR = 0x0010004D,
355. // Endpoint 'any' cannot be specified for a tunnel-mode rule.
356. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DTE\_VER = 0x0010004E,
357. // The target schema version does not support dynamic endpoints.
358. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DTE\_MISMATCH\_ADDR = 0x0010004F,
359. // When specifying tunnel endpoints in both IPv4 and IPv6, a tunnel
360. // endpoint may not be dynamic for one address family and explicit
361. // for the other. (A dynamic tunnel endpoint is one set to "Any".)
362. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PROFILE = 0x00100050,
363. // The profile type is invalid.
364. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ICMP = 0x00100060,
365. // Mismatch in number of ICMP and ICMP buffer.
366. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ICMP\_CODE = 0x00100061,
367. // Invalid ICMP code specified.
368. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_IF\_ID = 0x00100070,
369. // Number of interfaces and interface buffers do not match.
370. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_IF\_TYPE = 0x00100071,
371. // The interface type is invalid.
372. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ACTION = 0x00100080,
373. // The action is invalid.
374. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ALLOW\_BYPASS = 0x00100081,
375. // Allow-Bypass action specified, but the rule does not meet
376. // allow-bypass criteria (inbound, authenticate/encrypt flags set,
377. // remote machine auth list specified)
378. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DO\_NOT\_SECURE = 0x00100082,
379. // If the action is Do Not Secure, the auth and crypto sets must be
380. // null.
381. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ACTION\_BLOCK\_IS\_ENCRYPTED\_SECURE = 0x00100083,
382. // Block action was specified in conjunction with require security
383. // or require encryption.
384. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_INCOMPATIBLE\_FLAG\_OR\_ACTION\_WITH\_SECURITY\_REALM = 0x00100084,
385. // Firewall Rules with security realm Id field would require authentication
386. // and encryption, and action should be Allow.
387. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DIR = 0x00100090,
388. // The direction is invalid.
389. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PROT = 0x001000A0,
390. // The protocol number is invalid.
391. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PROT\_PROP = 0x001000A1,
392. // The protocol-specific options do not match the protocol that was
393. // chosen.
394. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DEFER\_EDGE\_PROP = 0x001000A2,
395. // The edge traversal flags are inconsistent. Defer To App must be
396. // set without Edge Traversal, but Defer To User must be set with
397. // Edge Traversal.
398. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ALLOW\_BYPASS\_OUTBOUND = 0x001000A3,
399. // Allow-Bypass action specified, but the rule does not meet
400. // allow-bypass criteria (authenticate/encrypt flags set)
401. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DEFER\_USER\_INVALID\_RULE = 0x001000A4,
402. // Defer to user' setting can only be used in a firewall rule where
403. // program path and TCP/UDP protocol are specified with no
404. // additional conditions.
405. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS = 0x001000B0,
406. // Invalid flags specified.
407. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTO\_AUTH = 0x001000B1,
408. // Autogenerate flag is set but Authenticate / Authenticate-encrypt
409. // flags are not set.
410. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTO\_BLOCK = 0x001000B2,
411. // Autogenerate flag is set but the action is block.
412. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTO\_DYN\_RPC = 0x001000B3,
413. // Autogenerate flag is set along with Dynamic RPC flag.
414. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTHENTICATE\_ENCRYPT = 0x001000B4,
415. // The Authentication and Authentication & Encryption flags cannot
416. // be used together.
417. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTH\_WITH\_ENC\_NEGOTIATE\_VER = 0x001000B5,
418. // The target schema version does not support Authentication
419. // (Dynamic Encryption).
420. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTH\_WITH\_ENC\_NEGOTIATE = 0x001000B6,
421. // When the Authentication (Dynamic Encryption) flag is set, the
422. // Authentication & Encryption flag must be set as well.
423. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_ESP\_NO\_ENCAP\_VER = 0x001000B7,
424. // The target schema version does not support Authentication (No
425. // Encapsulation).
426. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_ESP\_NO\_ENCAP = 0x001000B8,
427. // When the Authentication (No Encapsulation) flag is set, the
428. // Authentication flag must be set as well.
429. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_TUNNEL\_AUTH\_MODES\_VER = 0x001000B9,
430. // The target schema version does not support tunnel authentication
431. // modes.
432. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_TUNNEL\_AUTH\_MODES = 0x001000BA,
433. // The target schema version does not support tunnel authentication
434. // modes.
435. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_IP\_HTTPS\_VER = 0x001000BB,
436. // The target schema version does not support the IP\_HTTPS keyword.
437. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_IP\_TLS\_VER = 0x001000BB,
438. // The target schema version does not support the IP\_TLS keyword.
439. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PORTRANGE\_VER = 0x001000BC,
440. // The target schema version does not support port ranges.
441. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_ADDRS\_TRAVERSE\_DEFER\_VER = 0x001000BD,
442. // The target schema version does not support dynamic edge
443. // traversal.
444. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTH\_WITH\_ENC\_NEGOTIATE\_OUTBOUND = 0x001000BE,
445. // The Authentication (Dynamic Encryption) flag cannot be used when
446. // direction is Outbound.
447. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_AUTHENTICATE\_WITH\_OUTBOUND\_BYPASS\_VER = 0x001000BF,
448. // The target schema version does not support outbound Allow-Bypass
449. // rules.
450. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_REMOTE\_AUTH\_LIST = 0x001000C0,
451. // Authorization lists can only be used if authentication is
452. // required on the rule.
453. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_REMOTE\_USER\_LIST = 0x001000C1,
454. // Remote user authorization can only be applied to inbound rules.
455. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LOCAL\_USER\_LIST = 0x001000C2,
456. // The authorized local user list may not be used in conjunction
457. // with a service SID.
458. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LUA\_VER = 0x001000C3,
459. // The target schema version does not support the authorized local
460. // user list.
461. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LOCAL\_USER\_OWNER = 0x001000C4,
462. // The local user owner field may not be used in conjunction with a
463. // service SID.
464. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LOCAL\_USER\_OWNER\_VER = 0x001000C5,
465. // The target schema version does not support the local user owner
466. // field.
467. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_LUA\_CONDITIONAL\_VER = 0x001000C6,
468. // The target schema version does not support the authorized local
469. // user list containing conditional aces (e.g. aces with claims).
470. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_SYSTEMOS\_GAMEOS = 0x001000C7,
471. // The Sytem OS Only and Game OS Only flags cannot
472. // be used together.
473. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_CORTANA\_VER = 0x001000C8,
474. // The Sytem OS Only and Game OS Only flags cannot
475. // be used together.
476. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_REMOTENAME = 0x001000C9,
477. // The Sytem OS Only and Game OS Only flags cannot
478. // be used together.
479. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_ALLOW\_PROFILE\_CROSSING\_VER = 0x001000D0,
480. // The target schema version does not support profile crossing.
481. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_LOCAL\_ONLY\_MAPPED\_VER = 0x001000D1,
482. // The target schema version does not support local only mapping.
483. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PLATFORM = 0x001000E0,
484. // Number of valid OS Platforms and the list of valid OS Platforms
485. // do not match
486. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PLATFORM\_OP\_VER = 0x001000E1,
487. // The target schema version does not support the platform operator
488. // specified.
489. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PLATFORM\_OP = 0x001000E2,
490. // One of the platform operators is invalid.
491. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DTE\_NOANY\_ADDR = 0x001000F0,
492. // The DTM flag requires at least one dynamic endpoint.
493. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TUNNEL\_EXEMPT\_WITH\_GATEWAY = 0x001000F1,
494. // A dynamic tunnel-mode exemption rule cannot have tunnel
495. // endpoints.
496. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TUNNEL\_EXEMPT\_VER = 0x001000F2,
497. // The target schema version does not support tunnel-mode
498. // exemptions.
499. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_ADDR\_KEYWORD\_VER = 0x001000F3,
500. // The target schema version does not support one or more of the
501. // address keywords given.
502. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_KEY\_MODULE\_VER = 0x001000F4,
503. // The target schema version does not support custom key module
504. // preferences.
505. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_APP\_CONTAINER\_PACKAGE\_ID = 0x00100100,
506. // The application package SID is invalid.
507. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_APP\_CONTAINER\_PACKAGE\_ID\_VER = 0x00100101,
508. // The target schema version does not support application package
509. // SIDs.
510. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TRUST\_TUPLE\_KEYWORD\_INCOMPATIBLE = 0x00100200,
511. // Logical endpoints (trust tuples) cannot be combined with
512. // specific addresses or ports.
513. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TRUST\_TUPLE\_KEYWORD\_INVALID = 0x00100201,
514. // One or more of the logical endpoints (trust tuples) are invalid.
515. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TRUST\_TUPLE\_KEYWORD\_VER = 0x00100202,
516. // The target schema version does not support logical endpoints
517. // (trust tuples).
518. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_INTERFACE\_TYPES\_VER = 0x00100301,
519. // The target schema version does not support the specified
520. // local interface type
521. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_NETNAMES\_VER = 0x00100401,
522. // The target schema version does not support the specified
523. // local interface type
524. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_SECURITY\_REALM\_ID\_VER = 0x00100402,
525. // The target schema version does not support security realm Id
526. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_SYSTEMOS\_GAMEOS\_VER = 0x00100403,
527. // The target schema version does not support specifying System OS or Game OS flag
528. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_DEVMODE\_VER = 0x00100404,
529. // The target schema version does not support specifying Development mode flag
530. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_REMOTE\_SERVERNAME\_VER = 0x00100405,
531. // The target schema version does not support specifying Remote Server Name
532. // attributes.
533. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FQBN\_VER = 0x00100406,
534. // The target schema version does not support specifying fqbn
535. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_COMPARTMENT\_ID\_VER = 0x00100407,
536. // The target schema version does not support specifying compartment Id
537. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_CALLOUT\_AND\_AUDIT\_VER = 0x00100408,
538. // The target schema version does not support specifying callout and audit flag
540. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_AUTH\_SET\_ID = 0x00100500,
541. // The phase 1 auth set ID must be specified.
542. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_SET\_ID = 0x00100510,
543. // The quick mode crypto set ID must be specified.
544. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_SET\_ID = 0x00100511,
545. // The main mode crypto set ID must be specified.
546. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_KEY\_MANAGER\_DICTATE\_VER = 0x00100512,
547. // The target schema version does not support the Key Manager
548. // Dictation flag.
549. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_KEY\_MANAGER\_NOTIFY\_VER = 0x00100513,
550. // The target schema version does not support the Key Manager
551. // Notification flag.
552. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TRANSPORT\_MACHINE\_AUTHZ\_VER = 0x00100514,
553. // The target schema version does not support transport rule
554. // machine authorization lists.
555. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TRANSPORT\_USER\_AUTHZ\_VER = 0x00100515,
556. // The target schema version does not support transport rule user
557. // authorization lists.
558. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TRANSPORT\_MACHINE\_AUTHZ\_ON\_TUNNEL = 0x00100516,
559. // Transport machine authorization SDDL specified on tunnel-mode
560. // rule.
561. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_TRANSPORT\_USER\_AUTHZ\_ON\_TUNNEL = 0x00100517,
562. // Transport user authorization SDDL specified on tunnel-mode rule.
563. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PER\_RULE\_AND\_GLOBAL\_AUTHZ = 0x00100518,
564. // The Apply Global Authorization flag cannot be used when a
565. // per-rule authorization list is also specified.
566. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_FLAGS\_SECURITY\_REALM = 0x00100519,
567. // The target schema version does not support security realm flag.
568. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_SET\_ID = 0x00101000,
569. // The Set ID was not specified.
570. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_IPSEC\_PHASE = 0x00101010,
571. // The IPsec phase is invalid.
572. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_EMPTY\_SUITES = 0x00101020,
573. // No suites specified in the set.
574. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_AUTH\_METHOD = 0x00101030,
575. // One of the phase 1 auth methods is invalid.
576. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_AUTH\_METHOD = 0x00101031,
577. // One of the phase 2 auth methods is invalid.
578. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_METHOD\_ANONYMOUS = 0x00101032,
579. // Anonymous cannot be the only authentication method.
580. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_METHOD\_DUPLICATE = 0x00101033,
581. // The same authentication method cannot be used more than once
582. // within a set.
583. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_METHOD\_VER = 0x00101034,
584. // The target schema version does not support one or more of the
585. // authentication methods given.
586. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_SUITE\_FLAGS = 0x00101040,
587. // Invalid auth suite flags specified.
588. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_HEALTH\_CERT = 0x00101041,
589. // Machine certificates can only be used in phase 2 auth if they
590. // are machine health certificates.
591. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_SIGNCERT\_VER = 0x00101042,
592. // The target schema version does not support the requested
593. // certificate signing algorithm.
594. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_INTERMEDIATE\_CA\_VER = 0x00101043,
595. // The target schema version does not support targeting
596. // Intermediate CA's.
597. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_MACHINE\_SHKEY = 0x00101050,
598. // Machine Preshared Key was selected as an authentication type,
599. // but no key string was specified.
600. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_CA\_NAME = 0x00101060,
601. // The certificate authority name is required, and must be
602. // formatted as an X.509 distinguished name.
603. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_MIXED\_CERTS = 0x00101061,
604. // Machine health certificates and regular certificates cannot both
605. // be proposed within the same authentication set.
606. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_NON\_CONTIGUOUS\_CERTS = 0x00101062,
607. // When specifying multiple certificate authentication proposals,
608. // all the certificate proposals with the same signing method must
609. // must be grouped together within the set.
610. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_MIXED\_CA\_TYPE\_IN\_BLOCK = 0x00101063,
611. // This error value is not used.
612. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_MACHINE\_USER\_AUTH = 0x00101070,
613. // Both machine and user auth cannot be proposed within the same
614. // authentication set.
615. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_VER = 0x00101071,
616. // The target schema version does not support certificate criteria.
617. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_VER\_MISMATCH = 0x00101072,
618. // Certificate criteria version does not match schema version.
619. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_RENEWAL\_HASH = 0x00101073,
620. // The certificate criteria are invalid. A thumbprint hash must be
621. // specified when FollowRenewal is used.
622. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_INVALID\_HASH = 0x00101074,
623. // The certificate criteria are invalid. The thumbprint hash is
624. // invalid.
625. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_INVALID\_EKU = 0x00101075,
626. // The certificate criteria are invalid. One or more of the EKU's
627. // are invalid.
628. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_INVALID\_NAME\_TYPE = 0x00101076,
629. // The certificate criteria are invalid. The name type is invalid.
630. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_INVALID\_NAME = 0x00101077,
631. // The certificate criteria are invalid. The subject name is not
632. // valid.
633. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_INVALID\_CRITERIA\_TYPE = 0x00101078,
634. // The certificate criteria are invalid. The criteria type flags
635. // are invalid.
636. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_CERT\_CRITERIA\_MISSING\_CRITERIA = 0x00101079,
637. // The certificate criteria are invalid. You need to specify at
638. // least one set of validation criteria and one set of selection
639. // criteria.
640. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PROXY\_SERVER = 0x00101080,
641. // The Kerberos proxy name must be a fully qualified domain name
642. // (FQDN). For example: kerbproxy.contoso.com
643. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_PROXY\_SERVER\_VER = 0x00101081,
644. // The target schema version does not support kerberos proxy
645. // servers.
646. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_NON\_DEFAULT\_ID = 0x00105000,
647. // The main mode crypto set ID should be the global main mode
648. // crypto set ID.
649. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_FLAGS = 0x00105001,
650. // The phase 1 crypto set flags are invalid.
651. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_TIMEOUT\_MINUTES = 0x00105002,
652. // The main mode lifetime, in minutes, is invalid.
653. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_TIMEOUT\_SESSIONS = 0x00105003,
654. // The main mode lifetime, in sessions, is invalid.
655. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_KEY\_EXCHANGE = 0x00105004,
656. // One of the main mode key exchange algorithms is invalid.
657. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_ENCRYPTION = 0x00105005,
658. // One of the main mode encryption algorithms is invalid.
659. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_HASH = 0x00105006,
660. // One of the main mode hash algorithms is invalid.
661. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_ENCRYPTION\_VER = 0x00105007,
662. // The target schema version does not support one of the main mode
663. // encryption algorithms chosen.
664. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_HASH\_VER = 0x00105008,
665. // The target schema version does not support one of the main mode
666. // hash algorithms chosen.
667. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE1\_CRYPTO\_KEY\_EXCH\_VER = 0x00105009,
668. // The target schema version does not support one of the main mode
669. // key exchange algorithms chosen.
670. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_PFS = 0x00105020,
671. // One of the quick mode key exchange algorithms is invalid.
672. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_PROTOCOL = 0x00105021,
673. // One of the quick mode encapsulation types is invalid.
674. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_ENCRYPTION = 0x00105022,
675. // One of the quick mode encryption algorithms is invalid.
676. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_HASH = 0x00105023,
677. // One of the quick mode hash algorithms is invalid.
678. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_TIMEOUT\_MINUTES = 0x00105024,
679. // The quick mode lifetime, in minutes, is invalid.
680. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_TIMEOUT\_KBYTES = 0x00105025,
681. // The quick mode lifetime, in kilobytes, is invalid.
682. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_ENCRYPTION\_VER = 0x00105026,
683. // The target schema version does not support one of the quick mode
684. // encryption algorithms chosen.
685. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_HASH\_VER = 0x00105027,
686. // The target schema version does not support one of the quick mode
687. // hash algorithms chosen.
688. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_PHASE2\_CRYPTO\_PFS\_VER = 0x00105028,
689. // The target schema version does not support one of the quick mode
690. // key exchange algorithms chosen.
691. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_CRYPTO\_ENCR\_HASH = 0x00105040,
692. // Either Encryption or Hash must be specified.
693. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_CRYPTO\_ENCR\_HASH\_COMPAT = 0x00105041,
694. // The encryption and hash algorithms specified are incompatible.
695. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_SCHEMA\_VERSION = 0x00105050,
696. // The target schema version specified is not supported.
697. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_OR\_AND\_CONDITIONS = 0x00106000,
698. // Malformed query: Mismatch in the number of ORed terms and the
699. // terms array
700. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_AND\_CONDITIONS = 0x00106001,
701. // Malformed query: Mismatch in the number of ANDed conditions and
702. // conditions array
703. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_CONDITION\_KEY = 0x00106002,
704. // Malformed query: Invalid confition match key
705. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_CONDITION\_MATCH\_TYPE = 0x00106003,
706. // Malformed query: Invalid condition match type
707. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_CONDITION\_DATA\_TYPE = 0x00106004,
708. // Malformed query: Invalid condition data type
709. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_CONDITION\_KEY\_AND\_DATA\_TYPE = 0x00106005,
710. // Malformed query: Invalid key and data type combination
711. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEYS\_PROTOCOL\_PORT = 0x00106006,
712. // Malformed query: Protocol condition present without a protocol
713. // condition
714. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_PROFILE = 0x00106007,
715. // Malformed query: Profile Key unavailable for this object type
716. // queried
717. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_STATUS = 0x00106008,
718. // Malformed query: Status Key unavailable for this object type
719. // queried
720. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_FILTERID = 0x00106009,
721. // Malformed query: FilterID Key unavailable for this object type
722. // queried
723. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_APP\_PATH = 0x00106010,
724. // Malformed query: Application Key unavailable for this object
725. // type queried
726. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_PROTOCOL = 0x00106011,
727. // Malformed query: Protocol Key unavailable for this object type
728. // queried
729. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_LOCAL\_PORT = 0x00106012,
730. // Malformed query: Local Port Key unavailable for this object type
731. // queried
732. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_REMOTE\_PORT = 0x00106013,
733. // Malformed query: Remote Port Key unavailable for this object
734. // type queried
735. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_QUERY\_KEY\_SVC\_NAME = 0x00106015,
736. // Malformed query: Service Name Key unavailable for this object
737. // type queried
738. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_REQUIRE\_IN\_CLEAR\_OUT\_ON\_TRANSPORT = 0x00107000,
739. // Authentication mode,"Require inbound and clear outbound" can
740. // only be set when using IPsec tunneling.
741. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_BYPASS\_TUNNEL\_IF\_SECURE\_ON\_TRANSPORT = 0x00107001,
742. // Bypass Tunnel If Secure may not be set on Transport-Mode rules.
743. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_NOENCAP\_ON\_TUNNEL = 0x00107002,
744. // Authentication (No Encapsulation) may not be used on tunnel-mode
745. // rules.
746. FW\_RULE\_STATUS\_SEMANTIC\_ERROR\_AUTH\_NOENCAP\_ON\_PSK = 0x00107003,
747. // Authentication (No Encapsulation) may not be used on rules that
748. // contain preshared keys.
749. FW\_RULE\_STATUS\_RUNTIME\_ERROR = 0x00200000,
750. // A runtime error occurred while trying to enforce the rule.
751. FW\_RULE\_STATUS\_RUNTIME\_ERROR\_PHASE1\_AUTH\_NOT\_FOUND = 0x00200001,
752. // The phase 1 authentication set was not found.
753. FW\_RULE\_STATUS\_RUNTIME\_ERROR\_PHASE2\_AUTH\_NOT\_FOUND = 0x00200002,
754. // The phase 2 authentication set was not found.
755. FW\_RULE\_STATUS\_RUNTIME\_ERROR\_PHASE2\_CRYPTO\_NOT\_FOUND = 0x00200003,
756. // The quick mode cryptographic set was not found.
757. FW\_RULE\_STATUS\_RUNTIME\_ERROR\_AUTH\_MCHN\_SHKEY\_MISMATCH = 0x00200004,
758. // A conflict was detected between the phase 1 and phase 2
759. // authentication sets. When preshared keys are used in phase 1,
760. // there cannot be a phase 2 authentication set.
761. FW\_RULE\_STATUS\_RUNTIME\_ERROR\_PHASE1\_CRYPTO\_NOT\_FOUND = 0x00200005,
762. // The main mode cryptographic set was not found.
763. FW\_RULE\_STATUS\_RUNTIME\_ERROR\_AUTH\_NOENCAP\_ON\_TUNNEL = 0x00200006,
764. // Authentication (No Encapsulation) cannot be specified on a
765. // tunnel-mode rule.
766. FW\_RULE\_STATUS\_RUNTIME\_ERROR\_AUTH\_NOENCAP\_ON\_PSK = 0x00200007,
767. // Authentication (No Encapsulation) cannot be specified on a rule
768. // that uses a preshared key.
769. FW\_RULE\_STATUS\_RUNTIME\_ERROR\_KEY\_MODULE\_AUTH\_MISMATCH = 0x00200008,
770. // The key module in the rule is incompatible with the
771. // authentication methods specified in the associated
772. // authentication sets.
773. FW\_RULE\_STATUS\_ERROR = FW\_RULE\_STATUS\_PARSING\_ERROR |FW\_RULE\_STATUS\_SEMANTIC\_ERROR |FW\_RULE\_STATUS\_RUNTIME\_ERROR,
774. // An error occurred.
775. FW\_RULE\_STATUS\_ALL = 0xFFFF0000
776. // Enumerate all rules, regardless of status.
777. } FW\_RULE\_STATUS;
778. //rule status bitflags
779. typedef enum \_tag\_FW\_RULE\_STATUS\_CLASS
780. {
781. FW\_RULE\_STATUS\_CLASS\_OK = FW\_RULE\_STATUS\_OK, // The rule was parsed successfully from the store
782. FW\_RULE\_STATUS\_CLASS\_PARTIALLY\_IGNORED = FW\_RULE\_STATUS\_PARTIALLY\_IGNORED, // The rule has fields that the service can successfully ignore
783. FW\_RULE\_STATUS\_CLASS\_IGNORED = FW\_RULE\_STATUS\_IGNORED, // The rule has a higher version that the service must ignore
784. FW\_RULE\_STATUS\_CLASS\_PARSING\_ERROR = FW\_RULE\_STATUS\_PARSING\_ERROR, // The rule failed to be parsed correctly
785. FW\_RULE\_STATUS\_CLASS\_SEMANTIC\_ERROR = FW\_RULE\_STATUS\_SEMANTIC\_ERROR, //There is a semantic error when considering the fields of the rule in conjunction
786. FW\_RULE\_STATUS\_CLASS\_RUNTIME\_ERROR = FW\_RULE\_STATUS\_RUNTIME\_ERROR, // There is a runtime error when the object is considered in conjuntion with other Policy Objects.
787. FW\_RULE\_STATUS\_CLASS\_ERROR = FW\_RULE\_STATUS\_ERROR, // An Error occurred
788. FW\_RULE\_STATUS\_CLASS\_ALL = FW\_RULE\_STATUS\_ALL // All the status. (Used to enum ALL the rules, regardless the status.)
789. } FW\_RULE\_STATUS\_CLASS;
790. typedef enum \_tag\_FW\_OBJECT\_CTRL\_FLAG
791. {
792. FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA = 0x0001, // Allow RPC to marshall the metadata pointer in the objects
793. } FW\_OBJECT\_CTRL\_FLAG;
794. typedef enum \_tag\_FW\_ENFORCEMENT\_STATE
795. {
796. FW\_ENFORCEMENT\_STATE\_INVALID,
797. FW\_ENFORCEMENT\_STATE\_FULL,
798. FW\_ENFORCEMENT\_STATE\_WF\_OFF\_IN\_PROFILE,
799. FW\_ENFORCEMENT\_STATE\_CATEGORY\_OFF,
800. FW\_ENFORCEMENT\_STATE\_DISABLED\_OBJECT,
801. FW\_ENFORCEMENT\_STATE\_INACTIVE\_PROFILE,
802. FW\_ENFORCEMENT\_STATE\_LOCAL\_ADDRESS\_RESOLUTION\_EMPTY,
803. FW\_ENFORCEMENT\_STATE\_REMOTE\_ADDRESS\_RESOLUTION\_EMPTY,
804. FW\_ENFORCEMENT\_STATE\_LOCAL\_PORT\_RESOLUTION\_EMPTY,
805. FW\_ENFORCEMENT\_STATE\_REMOTE\_PORT\_RESOLUTION\_EMPTY,
806. FW\_ENFORCEMENT\_STATE\_INTERFACE\_RESOLUTION\_EMPTY,
807. FW\_ENFORCEMENT\_STATE\_APPLICATION\_RESOLUTION\_EMPTY,
808. FW\_ENFORCEMENT\_STATE\_REMOTE\_MACHINE\_EMPTY,
809. FW\_ENFORCEMENT\_STATE\_REMOTE\_USER\_EMPTY,
810. FW\_ENFORCEMENT\_STATE\_LOCAL\_GLOBAL\_OPEN\_PORTS\_DISALLOWED,
811. FW\_ENFORCEMENT\_STATE\_LOCAL\_AUTHORIZED\_APPLICATIONS\_DISALLOWED,
812. FW\_ENFORCEMENT\_STATE\_LOCAL\_FIREWALL\_RULES\_DISALLOWED,
813. FW\_ENFORCEMENT\_STATE\_LOCAL\_CONSEC\_RULES\_DISALLOWED,
814. FW\_ENFORCEMENT\_STATE\_MISMATCHED\_PLATFORM,
815. FW\_ENFORCEMENT\_STATE\_OPTIMIZED\_OUT,
816. FW\_ENFORCEMENT\_STATE\_LOCAL\_USER\_EMPTY,
817. FW\_ENFORCEMENT\_STATE\_TRANSPORT\_MACHINE\_SD\_EMPTY,
818. FW\_ENFORCEMENT\_STATE\_TRANSPORT\_USER\_SD\_EMPTY,
819. FW\_ENFORCEMENT\_STATE\_TUPLE\_RESOLUTION\_EMPTY,
820. FW\_ENFORCEMENT\_STATE\_NETNAME\_RESOLUTION\_EMPTY,
821. FW\_ENFORCEMENT\_STATE\_MAX
822. } FW\_ENFORCEMENT\_STATE;
823. typedef struct \_tag\_FW\_OBJECT\_METADATA
824. {
825. UINT64 qwFilterContextID;
826. [range(0, 100)]
827. DWORD dwNumEntries;
828. [size\_is(dwNumEntries)]
829. FW\_ENFORCEMENT\_STATE \*pEnforcementStates;
830. } FW\_OBJECT\_METADATA, \*PFW\_OBJECT\_METADATA;
831. typedef enum \_tag\_FW\_OS\_PLATFORM\_OP
832. {
833. FW\_OS\_PLATFORM\_OP\_EQ,
834. FW\_OS\_PLATFORM\_OP\_GTEQ,
835. FW\_OS\_PLATFORM\_OP\_MAX,
836. FW\_OS\_PLATFORM\_OP\_FIELD\_SIZE = 5,
837. FW\_OS\_PLATFORM\_OP\_FIELD\_MASK = 0xF8
838. } FW\_OS\_PLATFORM\_OP;
839. // Values for platform, major and minor versions correspond to values in the OSVERSIONINFOEX structure
840. typedef struct \_tag\_FW\_OS\_PLATFORM
841. {
842. BYTE bPlatform;
843. BYTE bMajorVersion;
844. BYTE bMinorVersion;
845. BYTE Reserved;
846. }FW\_OS\_PLATFORM, \*PFW\_OS\_PLATFORM;
847. typedef struct \_tag\_FW\_OS\_PLATFORM\_LIST
848. {
849. [range(0, 10000)]
850. DWORD dwNumEntries;
851. [size\_is(dwNumEntries)]
852. PFW\_OS\_PLATFORM pPlatforms;
853. }FW\_OS\_PLATFORM\_LIST, \*PFW\_OS\_PLATFORM\_LIST;
854. typedef struct \_tag\_FW\_NETWORK\_NAMES
855. {
856. DWORD dwNumEntries;
857. [string, unique, size\_is(dwNumEntries,)]
858. LPWSTR \*wszNames;
859. } FW\_NETWORK\_NAMES, \*PFW\_NETWORK\_NAMES;
860. typedef enum \_tag\_FW\_RULE\_ORIGIN\_TYPE
861. {
862. FW\_RULE\_ORIGIN\_INVALID,
863. FW\_RULE\_ORIGIN\_LOCAL,
864. FW\_RULE\_ORIGIN\_GP,
865. FW\_RULE\_ORIGIN\_DYNAMIC,
866. FW\_RULE\_ORIGIN\_AUTOGEN,
867. FW\_RULE\_ORIGIN\_HARDCODED,
868. FW\_RULE\_ORIGIN\_MAX
869. }FW\_RULE\_ORIGIN\_TYPE;
870. typedef enum \_tag\_FW\_ENUM\_RULES\_FLAGS
871. {
872. FW\_ENUM\_RULES\_FLAG\_NONE = 0x0000,
873. FW\_ENUM\_RULES\_FLAG\_RESOLVE\_NAME = 0x0001, // Resolves rule name if in the format of '@file.dll,-<resID>'
874. FW\_ENUM\_RULES\_FLAG\_RESOLVE\_DESCRIPTION = 0x0002, // Resolves rule descriptions if in the format of '@file.dll,-<resID>'
875. FW\_ENUM\_RULES\_FLAG\_RESOLVE\_APPLICATION = 0x0004, // Resolves environment variables in the application string
876. FW\_ENUM\_RULES\_FLAG\_RESOLVE\_KEYWORD = 0x0008, // Resolves Keywords in addresses and ports to the actual addresses and ports (dynamic store only)
877. FW\_ENUM\_RULES\_FLAG\_RESOLVE\_GPO\_NAME = 0x0010, // Resolves GPO name for the GP\_RSOP rules
878. FW\_ENUM\_RULES\_FLAG\_EFFECTIVE = 0x0020, // Enum Rules only if we attempted to push them to BFE (dynamic store only)
879. FW\_ENUM\_RULES\_FLAG\_INCLUDE\_METADATA = 0x0040, // Inlude Object MetaData in the Enumerated Object.
880. FW\_ENUM\_RULES\_FLAG\_MAX = 0x0080
881. }FW\_ENUM\_RULES\_FLAGS;
882. //ordered by priority - highest on top
883. typedef enum \_tag\_FW\_RULE\_ACTION
884. {
885. FW\_RULE\_ACTION\_INVALID = 0,
886. FW\_RULE\_ACTION\_ALLOW\_BYPASS,
887. FW\_RULE\_ACTION\_BLOCK,
888. FW\_RULE\_ACTION\_ALLOW,
889. FW\_RULE\_ACTION\_MAX
890. } FW\_RULE\_ACTION;
891. typedef enum \_tag\_FW\_RULE\_FLAGS
892. {
893. FW\_RULE\_FLAGS\_NONE = 0x0000,
894. FW\_RULE\_FLAGS\_ACTIVE = 0x0001,
895. FW\_RULE\_FLAGS\_AUTHENTICATE = 0x0002,
896. FW\_RULE\_FLAGS\_AUTHENTICATE\_WITH\_ENCRYPTION = 0x0004,
897. FW\_RULE\_FLAGS\_ROUTEABLE\_ADDRS\_TRAVERSE = 0x0008,
898. FW\_RULE\_FLAGS\_LOOSE\_SOURCE\_MAPPED = 0x00010,
899. FW\_RULE\_FLAGS\_MAX\_V2\_1 = 0x0020,
900. // This is the new "NoEncapsulation" flag in Windows 7 and Windows Server 2008 R2.
901. FW\_RULE\_FLAGS\_AUTH\_WITH\_NO\_ENCAPSULATION = 0x0020,
902. FW\_RULE\_FLAGS\_MAX\_V2\_9 = 0x0040,
903. // These are the new flags added for SSP in Windows 7 and Windows Server 2008 R2.
904. FW\_RULE\_FLAGS\_AUTH\_WITH\_ENC\_NEGOTIATE = 0x0040,
905. FW\_RULE\_FLAGS\_ROUTEABLE\_ADDRS\_TRAVERSE\_DEFER\_APP = 0x0080,
906. FW\_RULE\_FLAGS\_ROUTEABLE\_ADDRS\_TRAVERSE\_DEFER\_USER = 0x0100,
907. FW\_RULE\_FLAGS\_AUTHENTICATE\_BYPASS\_OUTBOUND = 0x0200,
908. FW\_RULE\_FLAGS\_MAX\_V2\_10 = 0x0400,
909. // This is the new flag in Windows 8 and Windows Server 2012 to allow profile crossings
910. // for clusters.
911. FW\_RULE\_FLAGS\_ALLOW\_PROFILE\_CROSSING = 0x0400,
912. // This is the new flag in Windows 8 and Windows Server 2012 to allow LOM on flows.
913. FW\_RULE\_FLAGS\_LOCAL\_ONLY\_MAPPED = 0x0800,
914. FW\_RULE\_FLAGS\_MAX\_V2\_20 = 0x1000,
915. FW\_RULE\_FLAGS\_LUA\_CONDITIONAL\_ACE = 0x1000,
916. FW\_RULE\_FLAGS\_BIND\_TO\_INTERFACE = 0x2000,
917. FW\_RULE\_FLAGS\_MAX = 0x4000,
918. }FW\_RULE\_FLAGS;
919. typedef enum \_tag\_FW\_RULE\_FLAGS2
920. {
921. FW\_RULE\_FLAGS2\_NONE = 0x0000,
922. FW\_RULE\_FLAGS2\_SYSTEMOS\_ONLY = 0x0001,
923. FW\_RULE\_FLAGS2\_GAMEOS\_ONLY = 0x0002,
924. FW\_RULE\_FLAGS2\_DEVMODE = 0x0004,
925. FW\_RULE\_FLAGS\_MAX\_V2\_26 = 0x0008,
926. FW\_RULE\_FLAGS2\_NOT\_USED\_VALUE\_8 = 0x0008,
927. FW\_RULE\_FLAGS2\_EMPTY\_REMOTENAME = 0x0010,
928. FW\_RULE\_FLAGS2\_NOT\_REMOTENAME = 0x0020,
929. FW\_RULE\_FLAGS2\_NOT\_USED\_VALUE\_64 = 0x0040,
930. FW\_RULE\_FLAGS2\_CALLOUT\_AND\_AUDIT = 0x0080,
931. FW\_RULE\_FLAGS2\_MAX = 0x0100
932. }FW\_RULE\_FLAGS2;
934. typedef struct \_tag\_FW\_RULE2\_0
935. {
936. struct \_tag\_FW\_RULE2\_0 \*pNext;
937. WORD wSchemaVersion;
938. [string, range(1,10001), ref]
939. WCHAR\* wszRuleId;
940. [string, range(1,10001)]
941. WCHAR\* wszName;
942. [string, range(1,10001)]
943. WCHAR\* wszDescription;
944. DWORD dwProfiles;
945. [range(FW\_DIR\_INVALID, FW\_DIR\_OUT)]
946. FW\_DIRECTION Direction;
947. [range(0,256)]
948. WORD wIpProtocol; //0-255 or FW\_IP\_PROTOCOL\_ANY
949. [switch\_type(WORD), switch\_is(wIpProtocol)]
950. union
951. {
952. // Ports specified if wIpProtocol = 6(TCP) or 17(UDP)
953. [case(6,17)]
954. struct
955. {
956. FW\_PORTS LocalPorts;
957. FW\_PORTS RemotePorts;
958. };
959. // ICMP types/codes specified if wIpProtocol = 1(ICMPv4) or 58(ICMPv6)
960. [case(1)]
961. FW\_ICMP\_TYPE\_CODE\_LIST V4TypeCodeList;
962. [case(58)]
963. FW\_ICMP\_TYPE\_CODE\_LIST V6TypeCodeList;
964. [default]
965. ;
966. };
967. FW\_ADDRESSES LocalAddresses;
968. FW\_ADDRESSES RemoteAddresses;
969. FW\_INTERFACE\_LUIDS LocalInterfaceIds;
970. DWORD dwLocalInterfaceTypes; // Bit flags from FW\_INTERFACE\_TYPE
971. [string, range(1,10001)]
972. WCHAR\* wszLocalApplication;
973. [string, range(1,10001)]
974. WCHAR\* wszLocalService;
975. [range(FW\_RULE\_ACTION\_INVALID, FW\_RULE\_ACTION\_MAX)]
976. FW\_RULE\_ACTION Action;
977. WORD wFlags; // Bit flags from FW\_RULE\_FLAGS
978. [string, range(1,10001)]
979. WCHAR\* wszRemoteMachineAuthorizationList; //Authorized remote machines SDDL
980. [string, range(1,10001)]
981. WCHAR\* wszRemoteUserAuthorizationList; //Authorized remote users SDDL
982. [string, range(1,10001)]
983. WCHAR\* wszEmbeddedContext;
984. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
986. FW\_RULE\_STATUS Status; //Parsing error if any, filled on return. On input, set this to FW\_RULE\_STATUS\_OK
987. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX)]
988. FW\_RULE\_ORIGIN\_TYPE Origin; //Rule origin, filled on enumerated rules. Ignored on input
989. [string, range(1,10001)]
990. WCHAR\* wszGPOName; //Name of originating GPO, if rule origin is GP.
991. DWORD Reserved;
992. } FW\_RULE2\_0, \*PFW\_RULE2\_0;
993. typedef struct \_tag\_FW\_RULE2\_10
994. {
995. struct \_tag\_FW\_RULE2\_10 \*pNext;
996. WORD wSchemaVersion;
997. [string, range(1,512), ref]
998. LPWSTR wszRuleId;
999. [string, range(1,10001)]
1000. LPWSTR wszName;
1001. [string, range(1,10001)]
1002. LPWSTR wszDescription;
1003. DWORD dwProfiles;
1004. [range(FW\_DIR\_INVALID, FW\_DIR\_OUT)]
1005. FW\_DIRECTION Direction;
1006. [range(0,256)]
1007. WORD wIpProtocol; //0-255 or FW\_IP\_PROTOCOL\_ANY
1008. [switch\_type(WORD), switch\_is(wIpProtocol)]
1009. union
1010. {
1011. // Ports specified if wIpProtocol = 6(TCP) or 17(UDP)
1012. [case(6,17)]
1013. struct
1014. {
1015. FW\_PORTS LocalPorts;
1016. FW\_PORTS RemotePorts;
1017. };
1018. // ICMP types/codes specified if wIpProtocol = 1(ICMPv4) or 58(ICMPv6)
1019. [case(1)]
1020. FW\_ICMP\_TYPE\_CODE\_LIST V4TypeCodeList;
1021. [case(58)]
1022. FW\_ICMP\_TYPE\_CODE\_LIST V6TypeCodeList;
1023. [default]
1024. ;
1025. };
1026. FW\_ADDRESSES LocalAddresses;
1027. FW\_ADDRESSES RemoteAddresses;
1028. FW\_INTERFACE\_LUIDS LocalInterfaceIds;
1029. DWORD dwLocalInterfaceTypes; // Bit flags from FW\_INTERFACE\_TYPE
1030. [string, range(1,10001)]
1031. LPWSTR wszLocalApplication;
1032. [string, range(1,10001)]
1033. LPWSTR wszLocalService;
1034. [range(FW\_RULE\_ACTION\_INVALID, FW\_RULE\_ACTION\_MAX)]
1035. FW\_RULE\_ACTION Action;
1036. WORD wFlags; // Bit flags from FW\_RULE\_FLAGS
1037. [string, range(1,10001)]
1038. LPWSTR wszRemoteMachineAuthorizationList; //Authorized remote machines SDDL
1039. [string, range(1,10001)]
1040. LPWSTR wszRemoteUserAuthorizationList; //Authorized remote users SDDL
1041. [string, range(1,10001)]
1042. LPWSTR wszEmbeddedContext;
1043. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
1045. FW\_RULE\_STATUS Status; //Parsing error if any, filled on return. On input, set this to FW\_RULE\_STATUS\_OK
1046. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX)]
1047. FW\_RULE\_ORIGIN\_TYPE Origin; //Rule origin, filled on enumerated rules. Ignored on input
1048. [string, range(1,10001)]
1049. LPWSTR wszGPOName; //Name of originating GPO, if rule origin is GP.
1050. DWORD Reserved;
1051. // [switch\_type(WORD), switch\_is(wBinaryVersion)]
1052. // union
1053. // {
1054. // [case(wBinaryVersion >= 0x0210)]
1055. // struct
1056. // {
1057. [size\_is((Reserved & FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA) ? 1 : 0)]
1058. PFW\_OBJECT\_METADATA pMetaData;
1059. // };
1060. // }; // End union wBinaryVersion
1061. } FW\_RULE2\_10, \*PFW\_RULE2\_10;
1062. typedef struct \_tag\_FW\_RULE2\_20
1063. {
1064. struct \_tag\_FW\_RULE2\_20 \*pNext;
1065. WORD wSchemaVersion;
1066. [string, range(1,512), ref]
1067. LPWSTR wszRuleId;
1068. [string, range(1,10001)]
1069. LPWSTR wszName;
1070. [string, range(1,10001)]
1071. LPWSTR wszDescription;
1072. DWORD dwProfiles;
1073. [range(FW\_DIR\_INVALID, FW\_DIR\_OUT)]
1074. FW\_DIRECTION Direction;
1075. [range(0,256)]
1076. WORD wIpProtocol; //0-255 or FW\_IP\_PROTOCOL\_ANY
1077. [switch\_type(WORD), switch\_is(wIpProtocol)]
1078. union
1079. {
1080. // Ports specified if wIpProtocol = 6(TCP) or 17(UDP)
1081. [case(6,17)]
1082. struct
1083. {
1084. FW\_PORTS LocalPorts;
1085. FW\_PORTS RemotePorts;
1086. };
1087. // ICMP types/codes specified if wIpProtocol = 1(ICMPv4) or 58(ICMPv6)
1088. [case(1)]
1089. FW\_ICMP\_TYPE\_CODE\_LIST V4TypeCodeList;
1090. [case(58)]
1091. FW\_ICMP\_TYPE\_CODE\_LIST V6TypeCodeList;
1092. [default]
1093. ;
1094. };
1095. FW\_ADDRESSES LocalAddresses;
1096. FW\_ADDRESSES RemoteAddresses;
1097. FW\_INTERFACE\_LUIDS LocalInterfaceIds;
1098. DWORD dwLocalInterfaceTypes; // Bit flags from FW\_INTERFACE\_TYPE
1099. [string, range(1,10001)]
1100. LPWSTR wszLocalApplication;
1101. [string, range(1,10001)]
1102. LPWSTR wszLocalService;
1103. [range(FW\_RULE\_ACTION\_INVALID, FW\_RULE\_ACTION\_MAX)]
1104. FW\_RULE\_ACTION Action;
1105. WORD wFlags; // Bit flags from FW\_RULE\_FLAGS
1106. [string, range(1,10001)]
1107. LPWSTR wszRemoteMachineAuthorizationList; //Authorized remote machines SDDL
1108. [string, range(1,10001)]
1109. LPWSTR wszRemoteUserAuthorizationList; //Authorized remote users SDDL
1110. [string, range(1,10001)]
1111. LPWSTR wszEmbeddedContext;
1112. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
1114. FW\_RULE\_STATUS Status; //Parsing error if any, filled on return. On input, set this to FW\_RULE\_STATUS\_OK
1115. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX)]
1116. FW\_RULE\_ORIGIN\_TYPE Origin; //Rule origin, filled on enumerated rules. Ignored on input
1117. [string, range(1,10001)]
1118. LPWSTR wszGPOName; //Name of originating GPO, if rule origin is GP.
1119. DWORD Reserved;
1120. // [switch\_type(WORD), switch\_is(wBinaryVersion)]
1121. // union
1122. // {
1123. // [case(wBinaryVersion >= 0x0210)]
1124. // struct
1125. // {
1126. [size\_is((Reserved & FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA) ? 1 : 0)]
1127. PFW\_OBJECT\_METADATA pMetaData;
1128. // };
1129. // }; // End union wBinaryVersion
1130. [string, range(1,10001)]
1131. WCHAR\* wszLocalUserAuthorizationList; //Authorized local users SDDL
1132. [string, range(1,10001)]
1133. WCHAR \* wszPackageId; // Application Container Package Id Sid
1134. [string, range(1,10001)]
1135. WCHAR \* wszLocalUserOwner; // User Owner of the Rule
1136. // Trust Tuple Keywords
1137. DWORD dwTrustTupleKeywords;
1138. } FW\_RULE2\_20, \*PFW\_RULE2\_20;
1139. typedef struct \_tag\_FW\_RULE2\_24
1140. {
1141. struct \_tag\_FW\_RULE2\_24 \*pNext;
1142. WORD wSchemaVersion;
1143. [string, range(1,512), ref]
1144. LPWSTR wszRuleId;
1145. [string, range(1,10001)]
1146. LPWSTR wszName;
1147. [string, range(1,10001)]
1148. LPWSTR wszDescription;
1149. DWORD dwProfiles;
1150. [range(FW\_DIR\_INVALID, FW\_DIR\_OUT)]
1151. FW\_DIRECTION Direction;
1152. [range(0,256)]
1153. WORD wIpProtocol; //0-255 or FW\_IP\_PROTOCOL\_ANY
1154. [switch\_type(WORD), switch\_is(wIpProtocol)]
1155. union
1156. {
1157. // Ports specified if wIpProtocol = 6(TCP) or 17(UDP)
1158. [case(6,17)]
1159. struct
1160. {
1161. FW\_PORTS LocalPorts;
1162. FW\_PORTS RemotePorts;
1163. };
1164. // ICMP types/codes specified if wIpProtocol = 1(ICMPv4) or 58(ICMPv6)
1165. [case(1)]
1166. FW\_ICMP\_TYPE\_CODE\_LIST V4TypeCodeList;
1167. [case(58)]
1168. FW\_ICMP\_TYPE\_CODE\_LIST V6TypeCodeList;
1169. [default]
1170. ;
1171. };
1172. FW\_ADDRESSES LocalAddresses;
1173. FW\_ADDRESSES RemoteAddresses;
1174. FW\_INTERFACE\_LUIDS LocalInterfaceIds;
1175. DWORD dwLocalInterfaceTypes; // Bit flags from FW\_INTERFACE\_TYPE
1176. [string, range(1,10001)]
1177. LPWSTR wszLocalApplication;
1178. [string, range(1,10001)]
1179. LPWSTR wszLocalService;
1180. [range(FW\_RULE\_ACTION\_INVALID, FW\_RULE\_ACTION\_MAX)]
1181. FW\_RULE\_ACTION Action;
1182. WORD wFlags; // Bit flags from FW\_RULE\_FLAGS
1183. [string, range(1,10001)]
1184. LPWSTR wszRemoteMachineAuthorizationList; //Authorized remote machines SDDL
1185. [string, range(1,10001)]
1186. LPWSTR wszRemoteUserAuthorizationList; //Authorized remote users SDDL
1187. [string, range(1,10001)]
1188. LPWSTR wszEmbeddedContext;
1189. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
1191. FW\_RULE\_STATUS Status; //Parsing error if any, filled on return. On input, set this to FW\_RULE\_STATUS\_OK
1192. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX)]
1193. FW\_RULE\_ORIGIN\_TYPE Origin; //Rule origin, filled on enumerated rules. Ignored on input
1194. [string, range(1,10001)]
1195. LPWSTR wszGPOName; //Name of originating GPO, if rule origin is GP.
1196. DWORD Reserved;
1197. // [switch\_type(WORD), switch\_is(wBinaryVersion)]
1198. // union
1199. // {
1200. // [case(wBinaryVersion >= 0x0210)]
1201. // struct
1202. // {
1203. [size\_is((Reserved & FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA) ? 1 : 0)]
1204. PFW\_OBJECT\_METADATA pMetaData;
1205. // };
1206. // }; // End union wBinaryVersion
1207. [string, range(1,10001)]
1208. WCHAR\* wszLocalUserAuthorizationList; //Authorized local users SDDL
1209. [string, range(1,10001)]
1210. WCHAR \* wszPackageId; // Application Container Package Id Sid
1211. [string, range(1,10001)]
1212. WCHAR \* wszLocalUserOwner; // User Owner of the Rule
1213. // Trust Tuple Keywords
1214. DWORD dwTrustTupleKeywords;
1215. FW\_NETWORK\_NAMES OnNetworkNames;
1216. [string, range(1,10001)]
1217. // security realm Id
1218. WCHAR\* wszSecurityRealmId; // Security Realm Id
1219. } FW\_RULE2\_24, \*PFW\_RULE2\_24;
1220. typedef struct \_tag\_FW\_RULE2\_25
1221. {
1222. struct \_tag\_FW\_RULE2\_25 \*pNext;
1223. WORD wSchemaVersion;
1224. [string, range(1,512), ref]
1225. LPWSTR wszRuleId;
1226. [string, range(1,10001)]
1227. LPWSTR wszName;
1228. [string, range(1,10001)]
1229. LPWSTR wszDescription;
1230. DWORD dwProfiles;
1231. [range(FW\_DIR\_INVALID, FW\_DIR\_OUT)]
1232. FW\_DIRECTION Direction;
1233. [range(0,256)]
1234. WORD wIpProtocol; //0-255 or FW\_IP\_PROTOCOL\_ANY
1235. [switch\_type(WORD), switch\_is(wIpProtocol)]
1236. union
1237. {
1238. // Ports specified if wIpProtocol = 6(TCP) or 17(UDP)
1239. [case(6,17)]
1240. struct
1241. {
1242. FW\_PORTS LocalPorts;
1243. FW\_PORTS RemotePorts;
1244. };
1245. // ICMP types/codes specified if wIpProtocol = 1(ICMPv4) or 58(ICMPv6)
1246. [case(1)]
1247. FW\_ICMP\_TYPE\_CODE\_LIST V4TypeCodeList;
1248. [case(58)]
1249. FW\_ICMP\_TYPE\_CODE\_LIST V6TypeCodeList;
1250. [default]
1251. ;
1252. };
1253. FW\_ADDRESSES LocalAddresses;
1254. FW\_ADDRESSES RemoteAddresses;
1255. FW\_INTERFACE\_LUIDS LocalInterfaceIds;
1256. DWORD dwLocalInterfaceTypes; // Bit flags from FW\_INTERFACE\_TYPE
1257. [string, range(1,10001)]
1258. LPWSTR wszLocalApplication;
1259. [string, range(1,10001)]
1260. LPWSTR wszLocalService;
1261. [range(FW\_RULE\_ACTION\_INVALID, FW\_RULE\_ACTION\_MAX)]
1262. FW\_RULE\_ACTION Action;
1263. WORD wFlags; // Bit flags from FW\_RULE\_FLAGS
1264. [string, range(1,10001)]
1265. LPWSTR wszRemoteMachineAuthorizationList; //Authorized remote machines SDDL
1266. [string, range(1,10001)]
1267. LPWSTR wszRemoteUserAuthorizationList; //Authorized remote users SDDL
1268. [string, range(1,10001)]
1269. LPWSTR wszEmbeddedContext;
1270. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
1272. FW\_RULE\_STATUS Status; //Parsing error if any, filled on return. On input, set this to FW\_RULE\_STATUS\_OK
1273. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX)]
1274. FW\_RULE\_ORIGIN\_TYPE Origin; //Rule origin, filled on enumerated rules. Ignored on input
1275. [string, range(1,10001)]
1276. LPWSTR wszGPOName; //Name of originating GPO, if rule origin is GP.
1277. DWORD Reserved;
1278. // [switch\_type(WORD), switch\_is(wBinaryVersion)]
1279. // union
1280. // {
1281. // [case(wBinaryVersion >= 0x0210)]
1282. // struct
1283. // {
1284. [size\_is((Reserved & FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA) ? 1 : 0)]
1285. PFW\_OBJECT\_METADATA pMetaData;
1286. // };
1287. // }; // End union wBinaryVersion
1288. [string, range(1,10001)]
1289. WCHAR\* wszLocalUserAuthorizationList; //Authorized local users SDDL
1290. [string, range(1,10001)]
1291. WCHAR \* wszPackageId; // Application Container Package Id Sid
1292. [string, range(1,10001)]
1293. WCHAR \* wszLocalUserOwner; // User Owner of the Rule
1294. // Trust Tuple Keywords
1295. DWORD dwTrustTupleKeywords;
1296. FW\_NETWORK\_NAMES OnNetworkNames;
1297. [string, range(1,10001)]
1298. // security realm Id
1299. WCHAR\* wszSecurityRealmId; // Security Realm Id
1300. WORD wFlags2; // Bit flags from FW\_RULE\_FLAGS2
1301. } FW\_RULE2\_25, \*PFW\_RULE2\_25;
1302. typedef struct \_tag\_FW\_RULE2\_26
1303. {
1304. struct \_tag\_FW\_RULE2\_26 \*pNext;
1305. WORD wSchemaVersion;
1306. [string, range(1,512), ref]
1307. LPWSTR wszRuleId;
1308. [string, range(1,10001)]
1309. LPWSTR wszName;
1310. [string, range(1,10001)]
1311. LPWSTR wszDescription;
1312. DWORD dwProfiles;
1313. [range(FW\_DIR\_INVALID, FW\_DIR\_OUT)]
1314. FW\_DIRECTION Direction;
1315. [range(0,256)]
1316. WORD wIpProtocol; //0-255 or FW\_IP\_PROTOCOL\_ANY
1317. [switch\_type(WORD), switch\_is(wIpProtocol)]
1318. union
1319. {
1320. // Ports specified if wIpProtocol = 6(TCP) or 17(UDP)
1321. [case(6,17)]
1322. struct
1323. {
1324. FW\_PORTS LocalPorts;
1325. FW\_PORTS RemotePorts;
1326. };
1327. // ICMP types/codes specified if wIpProtocol = 1(ICMPv4) or 58(ICMPv6)
1328. [case(1)]
1329. FW\_ICMP\_TYPE\_CODE\_LIST V4TypeCodeList;
1330. [case(58)]
1331. FW\_ICMP\_TYPE\_CODE\_LIST V6TypeCodeList;
1332. [default]
1333. ;
1334. };
1335. FW\_ADDRESSES LocalAddresses;
1336. FW\_ADDRESSES RemoteAddresses;
1337. FW\_INTERFACE\_LUIDS LocalInterfaceIds;
1338. DWORD dwLocalInterfaceTypes; // Bit flags from FW\_INTERFACE\_TYPE
1339. [string, range(1,10001)]
1340. LPWSTR wszLocalApplication;
1341. [string, range(1,10001)]
1342. LPWSTR wszLocalService;
1343. [range(FW\_RULE\_ACTION\_INVALID, FW\_RULE\_ACTION\_MAX)]
1344. FW\_RULE\_ACTION Action;
1345. WORD wFlags; // Bit flags from FW\_RULE\_FLAGS
1346. [string, range(1,10001)]
1347. LPWSTR wszRemoteMachineAuthorizationList; //Authorized remote machines SDDL [string, range(1,10001)]
1348. LPWSTR wszRemoteUserAuthorizationList; //Authorized remote users SDDL
1349. [string, range(1,10001)]
1350. LPWSTR wszEmbeddedContext;
1351. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
1353. FW\_RULE\_STATUS Status; //Parsing error if any, filled on return. On input, set this to FW\_RULE\_STATUS\_OK
1354. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX)]
1355. FW\_RULE\_ORIGIN\_TYPE Origin; //Rule origin, filled on enumerated rules. Ignored on input
1356. [string, range(1,10001)]
1357. LPWSTR wszGPOName; //Name of originating GPO, if rule origin is GP.
1358. DWORD Reserved;
1359. // [switch\_type(WORD), switch\_is(wBinaryVersion)]
1360. // union
1361. // {
1362. // [case(wBinaryVersion >= 0x0210)]
1363. // struct
1364. // {
1365. [size\_is((Reserved & FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA) ? 1 : 0)]
1366. PFW\_OBJECT\_METADATA pMetaData;
1367. // };
1368. // }; // End union wBinaryVersion
1369. [string, range(1,10001)]
1370. WCHAR\* wszLocalUserAuthorizationList; //Authorized local users SDDL
1371. [string, range(1,10001)]
1372. WCHAR \* wszPackageId; // Application Container Package Id Sid
1373. [string, range(1,10001)]
1374. WCHAR \* wszLocalUserOwner; // User Owner of the Rule
1375. // Trust Tuple Keywords
1376. DWORD dwTrustTupleKeywords;
1377. FW\_NETWORK\_NAMES OnNetworkNames;
1378. [string, range(1,10001)]
1379. // security realm Id
1380. WCHAR\* wszSecurityRealmId; // Security Realm Id
1381. WORD wFlags2; // Bit flags from FW\_RULE\_FLAGS2
1382. FW\_NETWORK\_NAMES RemoteOutServerNames;
1383. } FW\_RULE2\_26, \*PFW\_RULE2\_26;
1384. typedef struct \_tag\_FW\_RULE
1385. {
1386. struct \_tag\_FW\_RULE \*pNext;
1387. WORD wSchemaVersion;
1388. [string, range(1,512), ref]
1389. LPWSTR wszRuleId;
1390. [string, range(1,10001)]
1391. LPWSTR wszName;
1392. [string, range(1,10001)]
1393. LPWSTR wszDescription;
1394. DWORD dwProfiles;
1395. [range(FW\_DIR\_INVALID, FW\_DIR\_OUT)]
1396. FW\_DIRECTION Direction;
1397. [range(0,256)]
1398. WORD wIpProtocol; //0-255 or FW\_IP\_PROTOCOL\_ANY
1399. [switch\_type(WORD), switch\_is(wIpProtocol)]
1400. union
1401. {
1402. // Ports specified if wIpProtocol = 6(TCP) or 17(UDP)
1403. [case(6,17)]
1404. struct
1405. {
1406. FW\_PORTS LocalPorts;
1407. FW\_PORTS RemotePorts;
1408. };
1409. // ICMP types/codes specified if wIpProtocol = 1(ICMPv4) or 58(ICMPv6)
1410. [case(1)]
1411. FW\_ICMP\_TYPE\_CODE\_LIST V4TypeCodeList;
1412. [case(58)]
1413. FW\_ICMP\_TYPE\_CODE\_LIST V6TypeCodeList;
1414. [default]
1415. ;
1416. };
1417. FW\_ADDRESSES LocalAddresses;
1418. FW\_ADDRESSES RemoteAddresses;
1419. FW\_INTERFACE\_LUIDS LocalInterfaceIds;
1420. DWORD dwLocalInterfaceTypes; // Bit flags from FW\_INTERFACE\_TYPE
1421. [string, range(1,10001)]
1422. LPWSTR wszLocalApplication;
1423. [string, range(1,10001)]
1424. LPWSTR wszLocalService;
1425. [range(FW\_RULE\_ACTION\_INVALID, FW\_RULE\_ACTION\_MAX)]
1426. FW\_RULE\_ACTION Action;
1427. WORD wFlags; // Bit flags from FW\_RULE\_FLAGS
1428. [string, range(1,10001)]
1429. LPWSTR wszRemoteMachineAuthorizationList; //Authorized remote machines SDDL
1430. [string, range(1,10001)]
1431. LPWSTR wszRemoteUserAuthorizationList; //Authorized remote users SDDL
1432. [string, range(1,10001)]
1433. LPWSTR wszEmbeddedContext;
1434. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
1436. FW\_RULE\_STATUS Status; //Parsing error if any, filled on return. On input, set this to FW\_RULE\_STATUS\_OK
1437. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX)]
1438. FW\_RULE\_ORIGIN\_TYPE Origin; //Rule origin, filled on enumerated rules. Ignored on input
1439. [string, range(1,10001)]
1440. LPWSTR wszGPOName; //Name of originating GPO, if rule origin is GP.
1441. DWORD Reserved;
1442. // [switch\_type(WORD), switch\_is(wBinaryVersion)]
1443. // union
1444. // {
1445. // [case(wBinaryVersion >= 0x0210)]
1446. // struct
1447. // {
1448. [size\_is((Reserved & FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA) ? 1 : 0)]
1449. PFW\_OBJECT\_METADATA pMetaData;
1450. // };
1451. // }; // End union wBinaryVersion
1452. [string, range(1,10001)]
1453. WCHAR\* wszLocalUserAuthorizationList; //Authorized local users SDDL
1454. [string, range(1,10001)]
1455. WCHAR \* wszPackageId; // Application Container Package Id Sid
1456. [string, range(1,10001)]
1457. WCHAR \* wszLocalUserOwner; // User Owner of the Rule
1458. // Trust Tuple Keywords
1459. DWORD dwTrustTupleKeywords;
1460. FW\_NETWORK\_NAMES OnNetworkNames;
1461. [string, range(1,10001)]
1462. // security realm Id
1463. WCHAR\* wszSecurityRealmId; // Security Realm Id
1464. WORD wFlags2; // Bit flags from FW\_RULE\_FLAGS2
1465. FW\_NETWORK\_NAMES RemoteOutServerNames;
1466. [string, range(1,10001)]
1467. WCHAR\* wszFqbn;
1468. DWORD compartmentId;
1469. } FW\_RULE, \*PFW\_RULE;
1470. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
1471. \* \*
1472. \* Configuration settings structures \*
1473. \* \*
1474. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
1475. #define FW\_PROFILE\_CONFIG\_LOG\_FILE\_SIZE\_MIN 1
1476. #define FW\_PROFILE\_CONFIG\_LOG\_FILE\_SIZE\_MAX 32767
1477. //All config settings are read-only for dynamic store
1478. typedef enum \_tag\_FW\_PROFILE\_CONFIG
1479. { // Type
1480. FW\_PROFILE\_CONFIG\_INVALID,
1481. FW\_PROFILE\_CONFIG\_ENABLE\_FW, // Boolean (as DWORD)
1482. FW\_PROFILE\_CONFIG\_DISABLE\_STEALTH\_MODE, // Boolean (as DWORD)
1483. FW\_PROFILE\_CONFIG\_SHIELDED, // Boolean (as DWORD)
1484. FW\_PROFILE\_CONFIG\_DISABLE\_UNICAST\_RESPONSES\_TO\_MULTICAST\_BROADCAST,
1485. // Boolean (as DWORD)
1486. FW\_PROFILE\_CONFIG\_LOG\_DROPPED\_PACKETS, // Boolean (as DWORD)
1487. FW\_PROFILE\_CONFIG\_LOG\_SUCCESS\_CONNECTIONS, // Boolean (as DWORD)
1488. FW\_PROFILE\_CONFIG\_LOG\_IGNORED\_RULES, // Boolean (as DWORD)
1489. FW\_PROFILE\_CONFIG\_LOG\_MAX\_FILE\_SIZE, // DWORD (in KBytes)
1490. FW\_PROFILE\_CONFIG\_LOG\_FILE\_PATH, // String
1491. FW\_PROFILE\_CONFIG\_DISABLE\_INBOUND\_NOTIFICATIONS, // Boolean (as DWORD)
1492. FW\_PROFILE\_CONFIG\_AUTH\_APPS\_ALLOW\_USER\_PREF\_MERGE, // Boolean (as DWORD) - GP\_RSOP/GPO store only
1493. FW\_PROFILE\_CONFIG\_GLOBAL\_PORTS\_ALLOW\_USER\_PREF\_MERGE, // Boolean (as DWORD) - GP\_RSOP/GPO store only
1494. FW\_PROFILE\_CONFIG\_ALLOW\_LOCAL\_POLICY\_MERGE, // Boolean (as DWORD) - GP\_RSOP/GPO store only
1495. FW\_PROFILE\_CONFIG\_ALLOW\_LOCAL\_IPSEC\_POLICY\_MERGE, // Boolean (as DWORD) - GP\_RSOP/GPO store only
1496. FW\_PROFILE\_CONFIG\_DISABLED\_INTERFACES, // PFW\_INTERFACE\_LUIDS - Local store only
1497. FW\_PROFILE\_CONFIG\_DEFAULT\_OUTBOUND\_ACTION, // DWORD(0 = Allow, 1 = block)
1498. FW\_PROFILE\_CONFIG\_DEFAULT\_INBOUND\_ACTION, // DWORD(0 = Allow, 1 = block)
1499. FW\_PROFILE\_CONFIG\_DISABLE\_STEALTH\_MODE\_IPSEC\_SECURED\_PACKET\_EXEMPTION,
1500. // Boolean (as DWORD)
1501. FW\_PROFILE\_CONFIG\_MAX
1502. } FW\_PROFILE\_CONFIG;
1503. typedef enum \_FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_VALUES
1504. {
1505. FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_NONE = 0x0000,
1506. FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_NEIGHBOR\_DISC = 0x0001,
1507. FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_ICMP = 0x0002,
1508. FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_ROUTER\_DISC = 0x0004,
1509. FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_NEIGHBOR\_DISC\_RFC =
1510. FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_NEIGHBOR\_DISC | FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_ROUTER\_DISC,
1511. FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_DHCP = 0x0008,
1512. FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_MAX = 0x0010
1513. }FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_VALUES;
1514. typedef enum \_FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING\_VALUES
1515. {
1516. FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING\_NONE = 0, // Preshared key is not encoded. Kept in its wide-char format.
1517. FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING\_UTF\_8,
1518. FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING\_MAX
1519. } FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING\_VALUES;
1520. typedef enum \_FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_VALUES
1521. {
1522. FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_NEVER = 0, // IPsec does not cross NAT boundaries
1523. FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_SERVER\_BEHIND\_NAT,
1524. FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_SERVER\_AND\_CLIENT\_BEHIND\_NAT,
1525. FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_MAX
1526. } FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_VALUES;
1527. #define FW\_GLOBAL\_CONFIG\_CRL\_CHECK\_MAX 2
1528. #define FW\_GLOBAL\_CONFIG\_SA\_IDLE\_TIME\_MAX 3600
1529. #define FW\_GLOBAL\_CONFIG\_SA\_IDLE\_TIME\_MIN 300
1530. typedef enum \_FW\_GLOBAL\_CONFIG\_ENABLE\_PACKET\_QUEUE\_FLAGS
1531. {
1532. FW\_GLOBAL\_CONFIG\_PACKET\_QUEUE\_NONE,
1533. FW\_GLOBAL\_CONFIG\_PACKET\_QUEUE\_INBOUND,
1534. FW\_GLOBAL\_CONFIG\_PACKET\_QUEUE\_FORWARD,
1535. FW\_GLOBAL\_CONFIG\_PACKET\_QUEUE\_MAX
1536. } FW\_GLOBAL\_CONFIG\_ENABLE\_PACKET\_QUEUE\_FLAGS;
1537. #define FW\_GLOBAL\_CONFIG\_PACKET\_QUEUE\_VALIDATION\_MASK 0x00000003
1538. //All config settings are read-only for dynamic store
1539. typedef enum \_tag\_FW\_GLOBAL\_CONFIG
1540. { // Type
1541. FW\_GLOBAL\_CONFIG\_INVALID,
1542. FW\_GLOBAL\_CONFIG\_POLICY\_VERSION\_SUPPORTED, // Policy version supported by the Firewall service
1543. FW\_GLOBAL\_CONFIG\_CURRENT\_PROFILE, // FW\_PROFILE\_TYPE (dynamic store only)
1544. FW\_GLOBAL\_CONFIG\_DISABLE\_STATEFUL\_FTP, // Boolean (as DWORD)
1545. FW\_GLOBAL\_CONFIG\_DISABLE\_STATEFUL\_PPTP, // Deprecated, Boolean as (DWORD)
1546. FW\_GLOBAL\_CONFIG\_SA\_IDLE\_TIME, // DWORD (300-3600 seconds)
1547. FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING, // DWORD (a value from FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING\_VALUES)
1548. FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT, // DWORD (bit-flags from FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_VALUES)
1549. // Max value: FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_MAX-1
1550. FW\_GLOBAL\_CONFIG\_CRL\_CHECK, // DWORD 0 - disables CRL checking
1551. // 1 - CRL checking is attempted and certificate validation fails only if the
1552. // certificate is revoked. Other failures that are encountered during CRL checking
1553. // (such as the revocation URL being unreachable) do not cause certificate validation to fail.
1554. // 2 - checking is required and that certificate validation fails if any error is encountered
1555. // during CRL processing.
1556. FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT, // FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_VALUES
1557. FW\_GLOBAL\_CONFIG\_POLICY\_VERSION, // Policy version
1558. FW\_GLOBAL\_CONFIG\_BINARY\_VERSION\_SUPPORTED, // Binary version supported by the Firewall Service (structures)
1559. FW\_GLOBAL\_CONFIG\_IPSEC\_TUNNEL\_REMOTE\_MACHINE\_AUTHORIZATION\_LIST, // May be zero-length to indicate that all machines or users are authorized or may contain
1560. FW\_GLOBAL\_CONFIG\_IPSEC\_TUNNEL\_REMOTE\_USER\_AUTHORIZATION\_LIST, // a null-terminated, Unicode string describing a security descriptor in SDDL.
1561. FW\_GLOBAL\_CONFIG\_OPPORTUNISTICALLY\_MATCH\_AUTH\_SET\_PER\_KM, // Boolean (as DWORD)
1562. FW\_GLOBAL\_CONFIG\_IPSEC\_TRANSPORT\_REMOTE\_MACHINE\_AUTHORIZATION\_LIST,
1563. FW\_GLOBAL\_CONFIG\_IPSEC\_TRANSPORT\_REMOTE\_USER\_AUTHORIZATION\_LIST,
1564. FW\_GLOBAL\_CONFIG\_ENABLE\_PACKET\_QUEUE,
1565. FW\_GLOBAL\_CONFIG\_MAX
1566. } FW\_GLOBAL\_CONFIG;
1567. typedef enum \_FW\_CONFIG\_FLAGS
1568. {
1569. FW\_CONFIG\_FLAG\_RETURN\_DEFAULT\_IF\_NOT\_FOUND = 0x0001 // If specified, if FWGetConfig or FWGetGlobalConfig fail to
1570. // find the configuration value in the store, the call will succeed and
1571. // return the default value used by the firewall service.
1572. // If not specified, if FWGetConfig or FWGetGlobalConfig fail to
1573. // find the configuration value in the store, the call will fail
1574. // with ERROR\_FILE\_NOT\_FOUND.
1575. } FW\_CONFIG\_FLAGS;
1576. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
1577. \* \*
1578. \* Network state structures.
1579. \* \*
1580. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
1581. // Based on INetwork (q.v.)
1582. typedef struct tag\_FW\_NETWORK
1583. {
1584. [string, unique]
1585. wchar\_t\* pszName;
1586. FW\_PROFILE\_TYPE ProfileType;
1587. } FW\_NETWORK, \*PFW\_NETWORK;
1588. // Adapter that can have the firewall enabled/disabled.
1589. typedef struct tag\_FW\_ADAPTER
1590. {
1591. [string, unique]
1592. wchar\_t\* pszFriendlyName;
1593. GUID Guid;
1594. } FW\_ADAPTER, \*PFW\_ADAPTER;
1595. typedef struct tag\_FW\_DIAG\_APP
1596. {
1597. [string, unique]
1598. wchar\_t\* pszAppPath;
1599. } FW\_DIAG\_APP, \*PFW\_DIAG\_APP;
1600. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
1601. \* \*
1602. \* Third-party firewall products structures.
1603. \* \*
1604. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
1605. // Different types of rules that the firewall supports.
1606. typedef
1607. [v1\_enum]
1608. enum tag\_FW\_RULE\_CATEGORY
1609. {
1610. FW\_RULE\_CATEGORY\_BOOT,
1611. FW\_RULE\_CATEGORY\_STEALTH,
1612. FW\_RULE\_CATEGORY\_FIREWALL,
1613. FW\_RULE\_CATEGORY\_CONSEC,
1614. // Not a valid rule category -- only used for bounds checking.
1615. FW\_RULE\_CATEGORY\_MAX
1616. } FW\_RULE\_CATEGORY, \*PFW\_RULE\_CATEGORY;
1617. // Struct representing a third-party firewall product.
1618. typedef struct tag\_FW\_PRODUCT
1619. {
1620. // Currently, no flags are defined, so this is just a placeholder.
1621. DWORD dwFlags;
1622. // Array of rule types implemented by the firewall. May be zero length in
1623. // which case branding is confirmed but Windows Firewall functionality is
1624. // not replaced.
1625. DWORD dwNumRuleCategories;
1626. [size\_is(dwNumRuleCategories), unique]
1627. FW\_RULE\_CATEGORY\* pRuleCategories;
1628. [string, ref]
1629. wchar\_t\* pszDisplayName;
1630. // The following field is only used when enumerating the registered
1631. // products. It must be null when calling FWRegisterProduct.
1632. [string, unique]
1633. wchar\_t\* pszPathToSignedProductExe;
1634. } FW\_PRODUCT, \*PFW\_PRODUCT;
1635. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
1636. \* \*
1637. \* Connection Security Rule structures \*
1638. \* \*
1639. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
1640. typedef enum \_tag\_FW\_IP\_VERSION
1641. {
1642. FW\_IP\_VERSION\_INVALID,
1643. FW\_IP\_VERSION\_V4,
1644. FW\_IP\_VERSION\_V6,
1645. FW\_IP\_VERSION\_MAX
1646. }FW\_IP\_VERSION;
1647. typedef enum \_tag\_FW\_IPSEC\_PHASE
1648. {
1649. FW\_IPSEC\_PHASE\_INVALID,
1650. FW\_IPSEC\_PHASE\_1,
1651. FW\_IPSEC\_PHASE\_2,
1652. FW\_IPSEC\_PHASE\_MAX
1653. }FW\_IPSEC\_PHASE;
1654. typedef enum \_tag\_FW\_CS\_RULE\_FLAGS
1655. {
1656. FW\_CS\_RULE\_FLAGS\_NONE = 0x00,
1657. FW\_CS\_RULE\_FLAGS\_ACTIVE = 0x01,
1658. FW\_CS\_RULE\_FLAGS\_DTM = 0x02,
1659. FW\_CS\_RULE\_FLAGS\_TUNNEL\_BYPASS\_IF\_ENCRYPTED = 0x08,
1660. FW\_CS\_RULE\_FLAGS\_OUTBOUND\_CLEAR = 0x10,
1661. FW\_CS\_RULE\_FLAGS\_APPLY\_AUTHZ = 0x20,
1662. FW\_CS\_RULE\_FLAGS\_KEY\_MANAGER\_ALLOW\_DICTATE\_KEY = 0x40,
1663. FW\_CS\_RULE\_FLAGS\_KEY\_MANAGER\_ALLOW\_NOTIFY\_KEY = 0x80,
1664. FW\_CS\_RULE\_FLAGS\_SECURITY\_REALM = 0x100,
1665. FW\_CS\_RULE\_FLAGS\_MAX = 0x200,
1666. FW\_CS\_RULE\_FLAGS\_MAX\_V2\_1 = 0x02,
1667. FW\_CS\_RULE\_FLAGS\_MAX\_V2\_8 = 0x04,
1668. FW\_CS\_RULE\_FLAGS\_MAX\_V2\_10 = 0x40,
1669. FW\_CS\_RULE\_FLAGS\_MAX\_V2\_20 = 0x100
1670. }FW\_CS\_RULE\_FLAGS;
1671. typedef enum \_tag\_FW\_CS\_RULE\_ACTION
1672. {
1673. FW\_CS\_RULE\_ACTION\_INVALID,
1674. FW\_CS\_RULE\_ACTION\_SECURE\_SERVER,
1675. FW\_CS\_RULE\_ACTION\_BOUNDARY,
1676. FW\_CS\_RULE\_ACTION\_SECURE,
1677. FW\_CS\_RULE\_ACTION\_DO\_NOT\_SECURE,
1678. FW\_CS\_RULE\_ACTION\_MAX
1679. }FW\_CS\_RULE\_ACTION;
1680. typedef struct \_tag\_FW\_CS\_RULE2\_0
1681. {
1682. struct \_tag\_FW\_CS\_RULE2\_0 \*pNext;
1683. WORD wSchemaVersion;
1684. [string, range(1,512), ref]
1685. WCHAR\* wszRuleId;
1686. [string, range(1,10001)]
1687. WCHAR\* wszName;
1688. [string, range(1,10001)]
1689. WCHAR\* wszDescription;
1690. DWORD dwProfiles;
1691. FW\_ADDRESSES Endpoint1;
1692. FW\_ADDRESSES Endpoint2;
1693. FW\_INTERFACE\_LUIDS LocalInterfaceIds;
1694. DWORD dwLocalInterfaceTypes; // Bit flags from FW\_INTERFACE\_TYPE
1695. DWORD dwLocalTunnelEndpointV4;
1696. BYTE LocalTunnelEndpointV6[16];
1697. DWORD dwRemoteTunnelEndpointV4;
1698. BYTE RemoteTunnelEndpointV6[16];
1700. FW\_PORTS Endpoint1Ports;
1701. FW\_PORTS Endpoint2Ports;
1702. [range(0,256)]
1703. WORD wIpProtocol;
1704. [string, range(1,255)]
1705. WCHAR\* wszPhase1AuthSet; // Set this to FW\_DEFAULT\_PHASE1\_AUTH\_SET to use the default
1706. [string, range(1,255)]
1707. WCHAR\* wszPhase2CryptoSet; // Set this to FW\_DEFAULT\_PHASE2\_CRYPTO\_SET to use the default
1708. [string, range(1,255)]
1709. WCHAR\* wszPhase2AuthSet; // If NULL, no phase 2 authentication is performed
1710. // Set this to FW\_DEFAULT\_PHASE2\_AUTH\_SET to use the default
1711. // Phase 1 crypto is global; Set Id unnecessary
1712. [range(FW\_CS\_RULE\_ACTION\_SECURE\_SERVER, FW\_CS\_RULE\_ACTION\_MAX)]
1713. FW\_CS\_RULE\_ACTION Action;
1714. WORD wFlags; // Bit flags from FW\_CS\_RULE\_FLAGS
1715. [string, range(1,10001)]
1716. WCHAR\* wszEmbeddedContext;
1717. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
1718. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX-1)]
1719. FW\_RULE\_ORIGIN\_TYPE Origin; //Rule origin, filled on enumerated rules. Ignored on input
1720. [string, range(1,10001)]
1721. WCHAR\* wszGPOName; //Name of originating GPO, if rule origin is GP.
1722. FW\_RULE\_STATUS Status; //Parsing error if any, filled on return. On input, set this to FW\_RULE\_STATUS\_OK
1723. }FW\_CS\_RULE2\_0, \*PFW\_CS\_RULE2\_0;
1724. typedef enum \_tag\_FW\_KEY\_MODULE\_
1725. {
1726. FW\_KEY\_MODULE\_DEFAULT = 0x0,
1727. FW\_KEY\_MODULE\_IKEv1 = 0x1,
1728. FW\_KEY\_MODULE\_AUTHIP = 0x2,
1729. FW\_KEY\_MODULE\_IKEv2 = 0x4,
1730. FW\_KEY\_MODULE\_MAX = 0x8
1731. } FW\_KEY\_MODULE;
1732. typedef struct \_tag\_FW\_CS\_RULE2\_10
1733. {
1734. struct \_tag\_FW\_CS\_RULE2\_10 \*pNext;
1735. WORD wSchemaVersion;
1736. [string, range(1,512), ref]
1737. WCHAR\* wszRuleId;
1738. [string, range(1,10001)]
1739. WCHAR\* wszName;
1740. [string, range(1,10001)]
1741. WCHAR\* wszDescription;
1742. DWORD dwProfiles;
1743. FW\_ADDRESSES Endpoint1;
1744. FW\_ADDRESSES Endpoint2;
1745. FW\_INTERFACE\_LUIDS LocalInterfaceIds;
1746. DWORD dwLocalInterfaceTypes; // Bit flags from FW\_INTERFACE\_TYPE
1747. DWORD dwLocalTunnelEndpointV4;
1748. BYTE LocalTunnelEndpointV6[16];
1749. DWORD dwRemoteTunnelEndpointV4;
1750. BYTE RemoteTunnelEndpointV6[16];
1752. FW\_PORTS Endpoint1Ports;
1753. FW\_PORTS Endpoint2Ports;
1754. [range(0,256)]
1755. WORD wIpProtocol;
1756. [string, range(1,255)]
1757. WCHAR\* wszPhase1AuthSet; // Set this to FW\_DEFAULT\_PHASE1\_AUTH\_SET to use the default
1758. [string, range(1,255)]
1759. WCHAR\* wszPhase2CryptoSet; // Set this to FW\_DEFAULT\_PHASE2\_CRYPTO\_SET to use the default
1760. [string, range(1,255)]
1761. WCHAR\* wszPhase2AuthSet; // If NULL, no phase 2 authentication is performed
1762. // Set this to FW\_DEFAULT\_PHASE2\_AUTH\_SET to use the default
1763. // Phase 1 crypto is global; Set Id unnecessary
1764. [range(FW\_CS\_RULE\_ACTION\_SECURE\_SERVER, FW\_CS\_RULE\_ACTION\_MAX)]
1765. FW\_CS\_RULE\_ACTION Action;
1766. WORD wFlags; // Bit flags from FW\_CS\_RULE\_FLAGS
1767. [string, range(1,10001)]
1768. WCHAR\* wszEmbeddedContext;
1769. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
1770. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX-1)]
1771. FW\_RULE\_ORIGIN\_TYPE Origin; //Rule origin, filled on enumerated rules. Ignored on input
1772. [string, range(1,10001)]
1773. WCHAR\* wszGPOName; //Name of originating GPO, if rule origin is GP.
1774. FW\_RULE\_STATUS Status; //Parsing error if any, filled on return. On input, set this to FW\_RULE\_STATUS\_OK
1775. [string, range(1,512)]
1776. WCHAR\* wszMMParentRuleId;
1777. DWORD Reserved;
1778. [size\_is((Reserved & FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA) ? 1 : 0)]
1779. PFW\_OBJECT\_METADATA pMetaData;
1780. }FW\_CS\_RULE2\_10, \*PFW\_CS\_RULE2\_10;
1781. typedef struct \_tag\_FW\_CS\_RULE
1782. {
1783. struct \_tag\_FW\_CS\_RULE \*pNext;
1784. WORD wSchemaVersion;
1785. [string, range(1,512), ref]
1786. WCHAR\* wszRuleId;
1787. [string, range(1,10001)]
1788. WCHAR\* wszName;
1789. [string, range(1,10001)]
1790. WCHAR\* wszDescription;
1791. DWORD dwProfiles;
1792. FW\_ADDRESSES Endpoint1;
1793. FW\_ADDRESSES Endpoint2;
1794. FW\_INTERFACE\_LUIDS LocalInterfaceIds;
1795. DWORD dwLocalInterfaceTypes; // Bit flags from FW\_INTERFACE\_TYPE
1796. DWORD dwLocalTunnelEndpointV4;
1797. BYTE LocalTunnelEndpointV6[16];
1798. DWORD dwRemoteTunnelEndpointV4;
1799. BYTE RemoteTunnelEndpointV6[16];
1801. FW\_PORTS Endpoint1Ports;
1802. FW\_PORTS Endpoint2Ports;
1803. [range(0,256)]
1804. WORD wIpProtocol;
1805. [string, range(1,255)]
1806. WCHAR\* wszPhase1AuthSet; // Set this to FW\_DEFAULT\_PHASE1\_AUTH\_SET to use the default
1807. [string, range(1,255)]
1808. WCHAR\* wszPhase2CryptoSet; // Set this to FW\_DEFAULT\_PHASE2\_CRYPTO\_SET to use the default
1809. [string, range(1,255)]
1810. WCHAR\* wszPhase2AuthSet; // If NULL, no phase 2 authentication is performed
1811. // Set this to FW\_DEFAULT\_PHASE2\_AUTH\_SET to use the default
1812. // Phase 1 crypto is global; Set Id unnecessary
1813. [range(FW\_CS\_RULE\_ACTION\_SECURE\_SERVER, FW\_CS\_RULE\_ACTION\_MAX)]
1814. FW\_CS\_RULE\_ACTION Action;
1815. WORD wFlags; // Bit flags from FW\_CS\_RULE\_FLAGS
1816. [string, range(1,10001)]
1817. WCHAR\* wszEmbeddedContext;
1818. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
1819. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX-1)]
1820. FW\_RULE\_ORIGIN\_TYPE Origin; //Rule origin, filled on enumerated rules. Ignored on input
1821. [string, range(1,10001)]
1822. WCHAR\* wszGPOName; //Name of originating GPO, if rule origin is GP.
1823. FW\_RULE\_STATUS Status; //Parsing error if any, filled on return. On input, set this to FW\_RULE\_STATUS\_OK
1824. [string, range(1,512)]
1825. WCHAR\* wszMMParentRuleId;
1826. DWORD Reserved;
1827. [size\_is((Reserved & FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA) ? 1 : 0)]
1828. PFW\_OBJECT\_METADATA pMetaData;
1829. [string, range(1,512)]
1830. WCHAR\* wszRemoteTunnelEndpointFqdn;
1831. FW\_ADDRESSES RemoteTunnelEndpoints;
1832. DWORD dwKeyModules;
1833. DWORD FwdPathSALifetime; //in seconds. Lifetime of SAs initiated by FWD path traffic
1834. [string, range(1,10001)]
1835. LPWSTR wszTransportMachineAuthzSDDL; // SDDL describing machine authorization
1836. [string, range(1,10001)]
1837. LPWSTR wszTransportUserAuthzSDDL; // SDDL describing user authorization
1838. }FW\_CS\_RULE, \*PFW\_CS\_RULE;
1839. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
1840. \* \*
1841. \* Ipsec Authentication Sets (Phase 1 and 2) structures \*
1842. \* \*
1843. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
1844. typedef enum \_tag\_FW\_AUTH\_METHOD
1845. {
1846. FW\_AUTH\_METHOD\_INVALID,
1847. FW\_AUTH\_METHOD\_ANONYMOUS, // Phase 1 and 2
1848. FW\_AUTH\_METHOD\_MACHINE\_KERB, // Phase 1 only
1849. FW\_AUTH\_METHOD\_MACHINE\_SHKEY, // Phase 1 (IKE) only
1850. FW\_AUTH\_METHOD\_MACHINE\_NTLM, // Phase 1 (AuthIp) only
1851. // If machine cert is specified for Phase2, it MUST be a health cert,
1852. // and no other authentication suites may be defined other than possibly anonymous.
1853. FW\_AUTH\_METHOD\_MACHINE\_CERT, // Phase 1 and 2
1854. FW\_AUTH\_METHOD\_USER\_KERB, // Phase 2 only
1855. FW\_AUTH\_METHOD\_USER\_CERT, // Phase 2 only
1856. FW\_AUTH\_METHOD\_USER\_NTLM, // Phase 2 only
1857. FW\_AUTH\_METHOD\_MACHINE\_RESERVED, // Phase 1 and 2
1858. FW\_AUTH\_METHOD\_USER\_RESERVED, // Phase 2
1859. FW\_AUTH\_METHOD\_MAX,
1860. FW\_AUTH\_METHOD\_MAX\_2\_10 = (FW\_AUTH\_METHOD\_USER\_NTLM + 1)
1861. }FW\_AUTH\_METHOD;
1862. typedef enum \_tag\_FW\_AUTH\_SUITE\_FLAGS
1863. {
1864. FW\_AUTH\_SUITE\_FLAGS\_NONE = 0x0000,
1865. // For Method = FW\_AUTH\_METHOD\_MACHINE\_CERT, Phase 1 only
1866. FW\_AUTH\_SUITE\_FLAGS\_CERT\_EXCLUDE\_CA\_NAME = 0x0001,
1867. // For Method = FW\_AUTH\_METHOD\_MACHINE\_CERT, Phase 1 and 2
1868. // For phase2, if Method = FW\_AUTH\_METHOD\_MACHINE\_CERT, this flag MUST be specified
1869. FW\_AUTH\_SUITE\_FLAGS\_HEALTH\_CERT = 0x0002,
1870. // For Method = FW\_AUTH\_METHOD\_MACHINE\_CERT (Phase 1 and 2),FW\_AUTH\_METHOD\_USER\_CERT
1871. FW\_AUTH\_SUITE\_FLAGS\_PERFORM\_CERT\_ACCOUNT\_MAPPING = 0x0004,
1872. FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA256 = 0x0008,
1873. FW\_AUTH\_SUITE\_FLAGS\_CERT\_SIGNING\_ECDSA384 = 0x0010,
1874. FW\_AUTH\_SUITE\_FLAGS\_MAX\_V2\_1 = 0x0020,
1875. FW\_AUTH\_SUITE\_FLAGS\_INTERMEDIATE\_CA = 0x0020,
1876. FW\_AUTH\_SUITE\_FLAGS\_MAX\_V2\_10 = 0x0040,
1877. FW\_AUTH\_SUITE\_FLAGS\_ALLOW\_PROXY = 0x0040,
1878. FW\_AUTH\_SUITE\_FLAGS\_MAX = 0x0080
1879. }FW\_AUTH\_SUITE\_FLAGS;
1880. typedef struct \_tag\_FW\_AUTH\_SUITE2\_10
1881. {
1882. [range(FW\_AUTH\_METHOD\_INVALID+1, FW\_AUTH\_METHOD\_MAX)]
1883. FW\_AUTH\_METHOD Method;
1884. WORD wFlags; // Bit-flags from FW\_AUTH\_SUITE\_FLAGS
1885. [switch\_type(FW\_AUTH\_METHOD), switch\_is(Method)]
1886. union
1887. {
1888. // For Method = FW\_AUTH\_METHOD\_MACHINE\_CERT
1889. // For Method = FW\_AUTH\_METHOD\_USER\_CERT
1890. [case(FW\_AUTH\_METHOD\_MACHINE\_CERT,FW\_AUTH\_METHOD\_USER\_CERT)]
1891. struct
1892. {
1893. [ref, string]
1894. WCHAR\* wszCAName;
1895. };
1896. // For Method = FW\_AUTH\_METHOD\_MACHINE\_SHKEY
1897. [case(FW\_AUTH\_METHOD\_MACHINE\_SHKEY)]
1898. struct
1899. {
1900. [ref, string]
1901. WCHAR\* wszSHKey;
1902. };
1903. [default]
1904. ;
1905. };
1906. }FW\_AUTH\_SUITE2\_10, \*PFW\_AUTH\_SUITE2\_10;
1907. typedef enum \_tag\_FW\_CERT\_CRITERIA\_NAME\_TYPE
1908. {
1909. FW\_CERT\_CRITERIA\_NAME\_NONE,
1910. FW\_CERT\_CRITERIA\_NAME\_DNS,
1911. FW\_CERT\_CRITERIA\_NAME\_UPN,
1912. FW\_CERT\_CRITERIA\_NAME\_RFC822,
1913. FW\_CERT\_CRITERIA\_NAME\_CN,
1914. FW\_CERT\_CRITERIA\_NAME\_OU,
1915. FW\_CERT\_CRITERIA\_NAME\_O,
1916. FW\_CERT\_CRITERIA\_NAME\_DC,
1917. FW\_CERT\_CRITERIA\_NAME\_MAX
1918. }FW\_CERT\_CRITERIA\_NAME\_TYPE;
1919. typedef enum \_tag\_FW\_CERT\_CRITERIA\_TYPE
1920. {
1921. FW\_CERT\_CRITERIA\_TYPE\_BOTH,
1922. FW\_CERT\_CRITERIA\_TYPE\_SELECTION,
1923. FW\_CERT\_CRITERIA\_TYPE\_VALIDATION,
1924. FW\_CERT\_CRITERIA\_TYPE\_MAX
1925. }FW\_CERT\_CRITERIA\_TYPE;
1926. typedef enum \_tag\_FW\_CERT\_CRITERIA\_FLAGS
1927. {
1928. FW\_AUTH\_CERT\_CRITERIA\_FLAGS\_NONE = 0x0000,
1929. FW\_AUTH\_CERT\_CRITERIA\_FLAGS\_FOLLOW\_RENEWAL = 0x0001,
1930. FW\_AUTH\_CERT\_CRITERIA\_FLAGS\_MAX = 0x0002
1931. }FW\_AUTH\_CERT\_CRITERIA\_FLAGS;
1932. typedef struct \_tag\_FW\_CERT\_CRITERIA
1933. {
1934. WORD wSchemaVersion;
1935. WORD wFlags;
1936. FW\_CERT\_CRITERIA\_TYPE CertCriteriaType;
1937. FW\_CERT\_CRITERIA\_NAME\_TYPE NameType;
1938. [string, unique]
1939. LPWSTR wszName;
1940. DWORD dwNumEku;
1941. [size\_is(dwNumEku), unique]
1942. LPSTR\* ppEku;
1943. [string, unique]
1944. LPWSTR wszHash;
1945. }FW\_CERT\_CRITERIA, \*PFW\_CERT\_CRITERIA;
1946. typedef struct \_tag\_FW\_AUTH\_SUITE
1947. {
1948. [range(FW\_AUTH\_METHOD\_INVALID+1, FW\_AUTH\_METHOD\_MAX)]
1949. FW\_AUTH\_METHOD Method;
1950. WORD wFlags; // Bit-flags from FW\_AUTH\_SUITE\_FLAGS
1951. [switch\_type(FW\_AUTH\_METHOD), switch\_is(Method)]
1952. union
1953. {
1954. // For Method = FW\_AUTH\_METHOD\_MACHINE\_CERT
1955. // For Method = FW\_AUTH\_METHOD\_USER\_CERT
1956. [case(FW\_AUTH\_METHOD\_MACHINE\_CERT,FW\_AUTH\_METHOD\_USER\_CERT)]
1957. struct
1958. {
1959. [ref, string]
1960. WCHAR\* wszCAName;
1961. [unique]
1962. PFW\_CERT\_CRITERIA pCertCriteria;
1963. };
1964. // For Method = FW\_AUTH\_METHOD\_MACHINE\_SHKEY
1965. [case(FW\_AUTH\_METHOD\_MACHINE\_SHKEY)]
1966. struct
1967. {
1968. [ref, string]
1969. WCHAR\* wszSHKey;
1970. };
1971. // For Method = FW\_AUTH\_METHOD\_MACHINE\_KERB
1972. // For Method = FW\_AUTH\_METHOD\_USER\_KERB
1973. [case(FW\_AUTH\_METHOD\_MACHINE\_KERB, FW\_AUTH\_METHOD\_USER\_KERB)]
1974. struct
1975. {
1976. [unique, string]
1977. WCHAR\* wszProxyServer;
1978. };
1979. [default]
1980. ;
1981. };
1982. }FW\_AUTH\_SUITE, \*PFW\_AUTH\_SUITE;
1983. typedef enum \_tag\_FW\_AUTH\_SET\_FLAGS
1984. {
1985. FW\_AUTH\_SET\_FLAGS\_NONE = 0x00,
1986. FW\_AUTH\_SET\_FLAGS\_MAX = 0x01,
1987. } FW\_AUTH\_SET\_FLAGS;
1988. typedef struct \_tag\_FW\_AUTH\_SET2\_10
1989. {
1990. struct \_tag\_FW\_AUTH\_SET2\_10\* pNext;
1991. WORD wSchemaVersion;
1992. [range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)]
1993. FW\_IPSEC\_PHASE IpSecPhase;
1994. [string, range(1,255), ref]
1995. WCHAR\* wszSetId; // To make this the default auth set, set the Id to FW\_DEFAULT\_PHASE1\_AUTH\_SET
1996. // or FW\_DEFAULT\_PHASE2\_AUTH\_SET as appropriate.
1997. [string, range(1,10001)]
1998. WCHAR\* wszName;
1999. [string, range(1,10001)]
2000. WCHAR\* wszDescription;
2001. [string, range(1,10001)]
2002. WCHAR\* wszEmbeddedContext;
2003. [range(0, 10000)]
2004. DWORD dwNumSuites;
2005. [size\_is(dwNumSuites)]
2006. PFW\_AUTH\_SUITE2\_10 pSuites;
2007. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX-1)]
2008. FW\_RULE\_ORIGIN\_TYPE Origin; //Rule origin, filled on enumerated rules. Ignored on input
2009. [string, range(1,10001)]
2010. WCHAR\* wszGPOName; //Name of originating GPO, if rule origin is GP.
2011. FW\_RULE\_STATUS Status; //Parsing error if any, filled on return. On input, set this to FW\_RULE\_STATUS\_OK
2012. DWORD dwAuthSetFlags;
2013. }FW\_AUTH\_SET2\_10, \*PFW\_AUTH\_SET2\_10;
2014. typedef struct \_tag\_FW\_AUTH\_SET
2015. {
2016. struct \_tag\_FW\_AUTH\_SET\* pNext;
2017. WORD wSchemaVersion;
2018. [range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)]
2019. FW\_IPSEC\_PHASE IpSecPhase;
2020. [string, range(1,255), ref]
2021. WCHAR\* wszSetId; // To make this the default auth set, set the Id to FW\_DEFAULT\_PHASE1\_AUTH\_SET
2022. // or FW\_DEFAULT\_PHASE2\_AUTH\_SET as appropriate.
2023. [string, range(1,10001)]
2024. WCHAR\* wszName;
2025. [string, range(1,10001)]
2026. WCHAR\* wszDescription;
2027. [string, range(1,10001)]
2028. WCHAR\* wszEmbeddedContext;
2029. [range(0, 10000)]
2030. DWORD dwNumSuites;
2031. [size\_is(dwNumSuites)]
2032. PFW\_AUTH\_SUITE pSuites;
2033. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX-1)]
2034. FW\_RULE\_ORIGIN\_TYPE Origin; //Rule origin, filled on enumerated rules. Ignored on input
2035. [string, range(1,10001)]
2036. WCHAR\* wszGPOName; //Name of originating GPO, if rule origin is GP.
2037. FW\_RULE\_STATUS Status; //Parsing error if any, filled on return. On input, set this to FW\_RULE\_STATUS\_OK
2038. DWORD dwAuthSetFlags; //Flags from FW\_AUTH\_SET\_FLAGS
2039. }FW\_AUTH\_SET, \*PFW\_AUTH\_SET;
2040. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
2041. \* \*
2042. \* Ipsec Crypto Set structures \*
2043. \* \*
2044. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
2045. typedef enum \_tag\_FW\_CRYPTO\_KEY\_EXCHANGE\_TYPE
2046. {
2047. FW\_CRYPTO\_KEY\_EXCHANGE\_NONE = 0, //When enumerating SAs, this value may be returned. Invalid for all other cases
2048. FW\_CRYPTO\_KEY\_EXCHANGE\_DH1,
2049. FW\_CRYPTO\_KEY\_EXCHANGE\_DH2,
2050. FW\_CRYPTO\_KEY\_EXCHANGE\_ECDH256,
2051. FW\_CRYPTO\_KEY\_EXCHANGE\_ECDH384,
2052. FW\_CRYPTO\_KEY\_EXCHANGE\_DH2048,
2053. FW\_CRYPTO\_KEY\_EXCHANGE\_DH24,
2054. FW\_CRYPTO\_KEY\_EXCHANGE\_MAX,
2055. FW\_CRYPTO\_KEY\_EXCHANGE\_DH14 = FW\_CRYPTO\_KEY\_EXCHANGE\_DH2048,
2056. FW\_CRYPTO\_KEY\_EXCHANGE\_MAX\_V2\_10 = FW\_CRYPTO\_KEY\_EXCHANGE\_DH24
2057. }FW\_CRYPTO\_KEY\_EXCHANGE\_TYPE;
2058. typedef enum \_tag\_FW\_CRYPTO\_ENCRYPTION\_TYPE
2059. {
2060. FW\_CRYPTO\_ENCRYPTION\_NONE,
2061. FW\_CRYPTO\_ENCRYPTION\_DES,
2062. FW\_CRYPTO\_ENCRYPTION\_3DES,
2063. FW\_CRYPTO\_ENCRYPTION\_AES128,
2064. FW\_CRYPTO\_ENCRYPTION\_AES192,
2065. FW\_CRYPTO\_ENCRYPTION\_AES256,
2066. FW\_CRYPTO\_ENCRYPTION\_AES\_GCM128,
2067. FW\_CRYPTO\_ENCRYPTION\_AES\_GCM192,
2068. FW\_CRYPTO\_ENCRYPTION\_AES\_GCM256,
2069. FW\_CRYPTO\_ENCRYPTION\_MAX,
2070. FW\_CRYPTO\_ENCRYPTION\_MAX\_V2\_0 = FW\_CRYPTO\_ENCRYPTION\_AES\_GCM128
2071. }FW\_CRYPTO\_ENCRYPTION\_TYPE;
2072. typedef enum \_tag\_FW\_CRYPTO\_HASH\_TYPE
2073. {
2074. FW\_CRYPTO\_HASH\_NONE,
2075. FW\_CRYPTO\_HASH\_MD5,
2076. FW\_CRYPTO\_HASH\_SHA1,
2077. FW\_CRYPTO\_HASH\_SHA256,
2078. FW\_CRYPTO\_HASH\_SHA384,
2079. FW\_CRYPTO\_HASH\_AES\_GMAC128,
2080. FW\_CRYPTO\_HASH\_AES\_GMAC192,
2081. FW\_CRYPTO\_HASH\_AES\_GMAC256,
2082. FW\_CRYPTO\_HASH\_MAX,
2083. FW\_CRYPTO\_HASH\_MAX\_V2\_0 = FW\_CRYPTO\_HASH\_SHA256
2084. }FW\_CRYPTO\_HASH\_TYPE;
2085. typedef enum \_tag\_FW\_CRYPTO\_PROTOCOL\_TYPE
2086. {
2087. FW\_CRYPTO\_PROTOCOL\_INVALID,
2088. FW\_CRYPTO\_PROTOCOL\_AH,
2089. FW\_CRYPTO\_PROTOCOL\_ESP,
2090. FW\_CRYPTO\_PROTOCOL\_BOTH,
2091. FW\_CRYPTO\_PROTOCOL\_AUTH\_NO\_ENCAP,
2092. FW\_CRYPTO\_PROTOCOL\_MAX,
2093. FW\_CRYPTO\_PROTOCOL\_MAX\_2\_1 = (FW\_CRYPTO\_PROTOCOL\_BOTH + 1)
2094. }FW\_CRYPTO\_PROTOCOL\_TYPE;
2095. typedef enum \_tag\_FW\_CRYPTO\_SET\_FLAGS
2096. {
2097. FW\_CRYPTO\_SET\_FLAGS\_NONE = 0x00,
2098. FW\_CRYPTO\_SET\_FLAGS\_MAX = 0x01,
2099. } FW\_CRYPTO\_SET\_FLAGS;
2100. typedef struct \_tag\_FW\_PHASE1\_CRYPTO\_SUITE
2101. {
2102. [range(FW\_CRYPTO\_KEY\_EXCHANGE\_NONE, FW\_CRYPTO\_KEY\_EXCHANGE\_MAX-1)]
2103. FW\_CRYPTO\_KEY\_EXCHANGE\_TYPE KeyExchange;
2104. [range(FW\_CRYPTO\_ENCRYPTION\_NONE+1, FW\_CRYPTO\_ENCRYPTION\_MAX-1)]
2105. FW\_CRYPTO\_ENCRYPTION\_TYPE Encryption;
2106. [range(FW\_CRYPTO\_HASH\_NONE+1, FW\_CRYPTO\_HASH\_MAX-1)]
2107. FW\_CRYPTO\_HASH\_TYPE Hash;
2108. DWORD dwP1CryptoSuiteFlags;
2109. }FW\_PHASE1\_CRYPTO\_SUITE, \*PFW\_PHASE1\_CRYPTO\_SUITE;
2110. typedef struct \_tag\_FW\_PHASE2\_CRYPTO\_SUITE
2111. {
2112. [range(FW\_CRYPTO\_PROTOCOL\_INVALID+1, FW\_CRYPTO\_PROTOCOL\_MAX-1)]
2113. FW\_CRYPTO\_PROTOCOL\_TYPE Protocol;
2114. FW\_CRYPTO\_HASH\_TYPE AhHash;
2115. FW\_CRYPTO\_HASH\_TYPE EspHash;
2116. FW\_CRYPTO\_ENCRYPTION\_TYPE Encryption;
2117. DWORD dwTimeoutMinutes;
2118. DWORD dwTimeoutKBytes;
2119. DWORD dwP2CryptoSuiteFlags;
2120. }FW\_PHASE2\_CRYPTO\_SUITE, \*PFW\_PHASE2\_CRYPTO\_SUITE;
2121. typedef enum \_tag\_FW\_PHASE1\_CRYPTO\_FLAGS
2122. {
2123. FW\_PHASE1\_CRYPTO\_FLAGS\_NONE = 0x00,
2124. FW\_PHASE1\_CRYPTO\_FLAGS\_DO\_NOT\_SKIP\_DH = 0x01,
2125. FW\_PHASE1\_CRYPTO\_FLAGS\_MAX = 0x02
2126. }FW\_PHASE1\_CRYPTO\_FLAGS;
2127. typedef enum \_tag\_FW\_PHASE2\_CRYPTO\_PFS
2128. {
2129. FW\_PHASE2\_CRYPTO\_PFS\_INVALID,
2130. FW\_PHASE2\_CRYPTO\_PFS\_DISABLE,
2131. FW\_PHASE2\_CRYPTO\_PFS\_PHASE1,
2132. FW\_PHASE2\_CRYPTO\_PFS\_DH1,
2133. FW\_PHASE2\_CRYPTO\_PFS\_DH2,
2134. FW\_PHASE2\_CRYPTO\_PFS\_DH2048,
2135. FW\_PHASE2\_CRYPTO\_PFS\_ECDH256,
2136. FW\_PHASE2\_CRYPTO\_PFS\_ECDH384,
2137. FW\_PHASE2\_CRYPTO\_PFS\_DH24,
2138. FW\_PHASE2\_CRYPTO\_PFS\_MAX,
2139. FW\_PHASE2\_CRYPTO\_PFS\_MAX\_V2\_10 = FW\_PHASE2\_CRYPTO\_PFS\_DH24
2140. }FW\_PHASE2\_CRYPTO\_PFS;
2141. typedef struct \_tag\_FW\_CRYPTO\_SET
2142. {
2143. struct \_tag\_FW\_CRYPTO\_SET\* pNext;
2144. WORD wSchemaVersion;
2145. [range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)]
2146. FW\_IPSEC\_PHASE IpSecPhase;
2147. [string, range(1,255), ref]
2148. WCHAR\* wszSetId; // For phase 1 crypto, this MUST be set to FW\_PHASE1\_CRYPTO\_SET (there can only be one phase 1 crypto set)
2149. // For phase 2 crypto, set this to FW\_DEFAULT\_PHASE2\_CRYPTO\_SET to make it the default.
2150. [string, range(1,10001)]
2151. WCHAR\* wszName;
2152. [string, range(1,10001)]
2153. WCHAR\* wszDescription;
2154. [string, range(1,10001)]
2155. WCHAR\* wszEmbeddedContext;
2156. [switch\_type(FW\_IPSEC\_PHASE), switch\_is(IpSecPhase)]
2157. union
2158. {
2159. [case(FW\_IPSEC\_PHASE\_1)]
2160. struct
2161. {
2162. WORD wFlags; // Bit-flags from FW\_PHASE1\_CRYPTO\_FLAGS
2163. [range(0, 10000)]
2164. DWORD dwNumPhase1Suites;
2165. [size\_is(dwNumPhase1Suites)]
2166. PFW\_PHASE1\_CRYPTO\_SUITE pPhase1Suites;
2167. DWORD dwTimeOutMinutes;
2168. DWORD dwTimeOutSessions;
2169. };
2170. [case(FW\_IPSEC\_PHASE\_2)]
2171. struct
2172. {
2173. FW\_PHASE2\_CRYPTO\_PFS Pfs;
2174. [range(0, 10000)]
2175. DWORD dwNumPhase2Suites;
2176. [size\_is(dwNumPhase2Suites)]
2177. PFW\_PHASE2\_CRYPTO\_SUITE pPhase2Suites;
2178. };
2179. };
2180. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX-1)]
2181. FW\_RULE\_ORIGIN\_TYPE Origin; //Rule origin, filled on enumerated rules. Ignored on input
2182. [string, range(1,10001)]
2183. WCHAR\* wszGPOName; //Name of originating GPO, if rule origin is GP.
2184. FW\_RULE\_STATUS Status; //Parsing error if any, filled on return. On input, set this to FW\_RULE\_STATUS\_OK
2185. DWORD dwCryptoSetFlags; //flags from FW\_CRYPTO\_SET\_FLAGS
2186. }FW\_CRYPTO\_SET, \*PFW\_CRYPTO\_SET;
2187. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
2188. \* \*
2189. \* SA structures (dynamic store only) \*
2190. \* \*
2191. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
2192. typedef struct \_tag\_FW\_BYTE\_BLOB
2193. {
2194. [range(0, 10000)]
2195. DWORD dwSize;
2196. [size\_is(dwSize)]
2197. BYTE\* Blob;
2198. }FW\_BYTE\_BLOB, \*PFW\_BYTE\_BLOB;
2199. typedef struct \_tag\_FW\_COOKIE\_PAIR
2200. {
2201. UINT64 Initiator;
2202. UINT64 Responder;
2203. }FW\_COOKIE\_PAIR, \*PFW\_COOKIE\_PAIR;
2204. typedef enum \_tag\_FW\_PHASE1\_KEY\_MODULE\_TYPE
2205. {
2206. FW\_PHASE1\_KEY\_MODULE\_INVALID,
2207. FW\_PHASE1\_KEY\_MODULE\_IKE,
2208. FW\_PHASE1\_KEY\_MODULE\_AUTH\_IP,
2209. FW\_PHASE1\_KEY\_MODULE\_MAX
2210. }FW\_PHASE1\_KEY\_MODULE\_TYPE;
2211. typedef struct \_tag\_FW\_CERT\_INFO
2212. {
2213. FW\_BYTE\_BLOB SubjectName;
2214. [range(FW\_AUTH\_SUITE\_FLAGS\_NONE, FW\_AUTH\_SUITE\_FLAGS\_MAX-1)]
2215. //only FW\_AUTH\_SUITE\_FLAGS\_HEALTH\_CERT and/or FW\_AUTH\_SUITE\_FLAGS\_PERFORM\_CERT\_ACCOUNT\_MAPPING will be set
2216. DWORD dwCertFlags; //bit-flags from FW\_AUTH\_SUITE\_FLAGS
2217. }FW\_CERT\_INFO, \*PFW\_CERT\_INFO;
2218. typedef struct \_tag\_FW\_AUTH\_INFO
2219. {
2220. [range(FW\_AUTH\_METHOD\_INVALID + 1, FW\_AUTH\_METHOD\_MAX)]
2221. FW\_AUTH\_METHOD AuthMethod;
2222. [switch\_type(FW\_AUTH\_METHOD), switch\_is(AuthMethod)]
2223. union
2224. {
2225. [case(FW\_AUTH\_METHOD\_MACHINE\_CERT,FW\_AUTH\_METHOD\_USER\_CERT)]
2226. //for auth method = cert
2227. struct
2228. {
2229. FW\_CERT\_INFO MyCert;
2230. FW\_CERT\_INFO PeerCert;
2231. };
2232. [case(FW\_AUTH\_METHOD\_MACHINE\_KERB,FW\_AUTH\_METHOD\_USER\_KERB,
2233. FW\_AUTH\_METHOD\_MACHINE\_RESERVED,FW\_AUTH\_METHOD\_USER\_RESERVED)]
2234. //for auth\_method = kerb
2235. struct
2236. {
2237. [string, range(1,10001)]
2238. WCHAR\* wszMyId;
2239. [string, range(1,10001)]
2240. WCHAR\* wszPeerId;
2241. };
2242. [default]
2243. ;
2244. };
2245. DWORD dwAuthInfoFlags;
2246. }FW\_AUTH\_INFO, \*PFW\_AUTH\_INFO;
2247. typedef struct \_tag\_FW\_ENDPOINTS
2248. {
2249. [range(FW\_IP\_VERSION\_INVALID+1, FW\_IP\_VERSION\_MAX-1)]
2250. FW\_IP\_VERSION IpVersion;
2251. DWORD dwSourceV4Address;
2252. DWORD dwDestinationV4Address;
2253. BYTE SourceV6Address[16];
2254. BYTE DestinationV6Address[16];
2255. }FW\_ENDPOINTS, \*PFW\_ENDPOINTS;
2256. typedef struct \_tag\_FW\_PHASE1\_SA\_DETAILS
2257. {
2258. UINT64 SaId;
2259. [range(FW\_PHASE1\_KEY\_MODULE\_INVALID+1, FW\_PHASE1\_KEY\_MODULE\_MAX-1)]
2260. FW\_PHASE1\_KEY\_MODULE\_TYPE KeyModuleType;
2261. FW\_ENDPOINTS Endpoints; //0 = Any
2262. FW\_PHASE1\_CRYPTO\_SUITE SelectedProposal;
2263. DWORD dwProposalLifetimeKBytes; //currently not supported
2264. DWORD dwProposalLifetimeMinutes;
2265. DWORD dwProposalMaxNumPhase2;
2266. FW\_COOKIE\_PAIR CookiePair;
2267. PFW\_AUTH\_INFO pFirstAuth; //First authentication - always present
2268. PFW\_AUTH\_INFO pSecondAuth; //First authentication - may be NULL
2269. DWORD dwP1SaFlags; // currently set to 0
2270. }FW\_PHASE1\_SA\_DETAILS, \*PFW\_PHASE1\_SA\_DETAILS;
2271. typedef enum \_tag\_FW\_PHASE2\_TRAFFIC\_TYPE
2272. {
2273. FW\_PHASE2\_TRAFFIC\_TYPE\_INVALID,
2274. FW\_PHASE2\_TRAFFIC\_TYPE\_TRANSPORT,
2275. FW\_PHASE2\_TRAFFIC\_TYPE\_TUNNEL,
2276. FW\_PHASE2\_TRAFFIC\_TYPE\_MAX
2277. }FW\_PHASE2\_TRAFFIC\_TYPE;
2278. typedef struct \_tag\_FW\_PHASE2\_SA\_DETAILS
2279. {
2280. UINT64 SaId;
2281. [range(FW\_DIR\_INVALID+1, FW\_DIR\_MAX-1)]
2282. FW\_DIRECTION Direction;
2284. FW\_ENDPOINTS Endpoints; //0 = Any
2285. WORD wLocalPort; //0 = Any
2286. WORD wRemotePort; //0 = Any
2287. WORD wIpProtocol; //0-255 or FW\_IP\_PROTOCOL\_ANY
2288. FW\_PHASE2\_CRYPTO\_SUITE SelectedProposal;
2289. FW\_PHASE2\_CRYPTO\_PFS Pfs;
2291. GUID TransportFilterId;
2292. DWORD dwP2SaFlags; // currently set to 0
2293. }FW\_PHASE2\_SA\_DETAILS, \*PFW\_PHASE2\_SA\_DETAILS;
2294. typedef
2295. [switch\_type(FW\_PROFILE\_CONFIG)]
2296. union \_FW\_PROFILE\_CONFIG\_VALUE
2297. {
2298. [case(FW\_PROFILE\_CONFIG\_LOG\_FILE\_PATH)]
2299. [string, range(1,10001)]
2300. WCHAR\* wszStr;
2301. [case(FW\_PROFILE\_CONFIG\_DISABLED\_INTERFACES)]
2302. PFW\_INTERFACE\_LUIDS pDisabledInterfaces;
2303. [case(FW\_PROFILE\_CONFIG\_ENABLE\_FW,
2304. FW\_PROFILE\_CONFIG\_DISABLE\_STEALTH\_MODE,
2305. FW\_PROFILE\_CONFIG\_SHIELDED,
2306. FW\_PROFILE\_CONFIG\_DISABLE\_UNICAST\_RESPONSES\_TO\_MULTICAST\_BROADCAST,
2307. FW\_PROFILE\_CONFIG\_LOG\_DROPPED\_PACKETS,
2308. FW\_PROFILE\_CONFIG\_LOG\_SUCCESS\_CONNECTIONS,
2309. FW\_PROFILE\_CONFIG\_LOG\_IGNORED\_RULES,
2310. FW\_PROFILE\_CONFIG\_LOG\_MAX\_FILE\_SIZE,
2311. FW\_PROFILE\_CONFIG\_DISABLE\_INBOUND\_NOTIFICATIONS,
2312. FW\_PROFILE\_CONFIG\_AUTH\_APPS\_ALLOW\_USER\_PREF\_MERGE,
2313. FW\_PROFILE\_CONFIG\_GLOBAL\_PORTS\_ALLOW\_USER\_PREF\_MERGE,
2314. FW\_PROFILE\_CONFIG\_ALLOW\_LOCAL\_POLICY\_MERGE,
2315. FW\_PROFILE\_CONFIG\_ALLOW\_LOCAL\_IPSEC\_POLICY\_MERGE,
2316. FW\_PROFILE\_CONFIG\_DEFAULT\_OUTBOUND\_ACTION,
2317. FW\_PROFILE\_CONFIG\_DEFAULT\_INBOUND\_ACTION,
2318. FW\_PROFILE\_CONFIG\_DISABLE\_STEALTH\_MODE\_IPSEC\_SECURED\_PACKET\_EXEMPTION)]
2319. DWORD\* pdwVal;
2320. }FW\_PROFILE\_CONFIG\_VALUE, \*PFW\_PROFILE\_CONFIG\_VALUE;
2321. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
2322. \* \*
2323. \* Main Mode Rule Structures \*
2324. \* \*
2325. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
2326. typedef struct \_tag\_FW\_MM\_RULE
2327. {
2328. struct \_tag\_FW\_MM\_RULE \*pNext;
2329. WORD wSchemaVersion;
2330. [string, range(1,512), ref]
2331. WCHAR\* wszRuleId;
2332. [string, range(1,10001)]
2333. WCHAR\* wszName;
2334. [string, range(1,10001)]
2335. WCHAR\* wszDescription;
2336. DWORD dwProfiles;
2337. FW\_ADDRESSES Endpoint1;
2338. FW\_ADDRESSES Endpoint2;
2339. [string, range(1,255)]
2340. WCHAR\* wszPhase1AuthSet; // Set this to FW\_DEFAULT\_PHASE1\_AUTH\_SET to use the default
2341. [string, range(1,255)]
2342. WCHAR\* wszPhase1CryptoSet; // Set this to FW\_DEFAULT\_PHASE1\_CRYPTO\_SET to use the default
2343. WORD wFlags; // Bit flags from FW\_CS\_RULE\_FLAGS
2344. [string, range(1,10001)]
2345. WCHAR\* wszEmbeddedContext;
2346. FW\_OS\_PLATFORM\_LIST PlatformValidityList;
2347. [range(FW\_RULE\_ORIGIN\_INVALID, FW\_RULE\_ORIGIN\_MAX-1)]
2348. FW\_RULE\_ORIGIN\_TYPE Origin; //Rule origin, filled on enumerated rules. Ignored on input
2349. [string, range(1,10001)]
2350. WCHAR\* wszGPOName; //Name of originating GPO, if rule origin is GP.
2351. FW\_RULE\_STATUS Status; //Parsing error if any, filled on return. On input, set this to FW\_RULE\_STATUS\_OK
2352. DWORD Reserved;
2353. [size\_is((Reserved & FW\_OBJECT\_CTRL\_FLAG\_INCLUDE\_METADATA) ? 1 : 0)]
2354. PFW\_OBJECT\_METADATA pMetaData;
2355. }FW\_MM\_RULE, \*PFW\_MM\_RULE;
2356. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
2357. \* \*
2358. \* Query Structures \*
2359. \* \*
2360. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
2361. typedef enum \_tag\_FW\_MATCH\_KEY
2362. {
2363. FW\_MATCH\_KEY\_PROFILE,
2364. FW\_MATCH\_KEY\_STATUS,
2365. FW\_MATCH\_KEY\_OBJECTID,
2366. FW\_MATCH\_KEY\_FILTERID,
2367. FW\_MATCH\_KEY\_APP\_PATH, //The APP Path
2368. FW\_MATCH\_KEY\_PROTOCOL,
2369. FW\_MATCH\_KEY\_LOCAL\_PORT,
2370. FW\_MATCH\_KEY\_REMOTE\_PORT,
2371. FW\_MATCH\_KEY\_GROUP,
2372. FW\_MATCH\_KEY\_SVC\_NAME,
2373. FW\_MATCH\_KEY\_DIRECTION,
2374. FW\_MATCH\_KEY\_LOCAL\_USER\_OWNER,
2375. FW\_MATCH\_KEY\_PACKAGE\_ID,
2376. FW\_MATCH\_KEY\_FQBN,
2377. FW\_MATCH\_KEY\_COMPARTMENT\_ID,
2378. FW\_MATCH\_KEY\_MAX
2379. }FW\_MATCH\_KEY;
2380. typedef enum \_tag\_FW\_DATA\_TYPE
2381. {
2382. FW\_DATA\_TYPE\_EMPTY,
2383. FW\_DATA\_TYPE\_UINT8,
2384. FW\_DATA\_TYPE\_UINT16,
2385. FW\_DATA\_TYPE\_UINT32,
2386. FW\_DATA\_TYPE\_UINT64,
2387. FW\_DATA\_TYPE\_UNICODE\_STRING
2388. }FW\_DATA\_TYPE;
2389. typedef struct \_tag\_FW\_MATCH\_VALUE
2390. {
2391. FW\_DATA\_TYPE type;
2392. [switch\_type(FW\_DATA\_TYPE), switch\_is(type)]
2393. union
2394. {
2395. [case(FW\_DATA\_TYPE\_UINT8)]
2396. UINT8 uInt8;
2397. [case(FW\_DATA\_TYPE\_UINT16)]
2398. UINT16 uInt16;
2399. [case(FW\_DATA\_TYPE\_UINT32)]
2400. UINT32 uInt32;
2401. [case(FW\_DATA\_TYPE\_UINT64)]
2402. UINT64 uInt64;
2403. [case(FW\_DATA\_TYPE\_UNICODE\_STRING)]
2404. struct
2405. {
2406. [string, range(1,10001)]
2407. LPWSTR wszString;
2408. };
2409. [case(FW\_DATA\_TYPE\_EMPTY)]
2410. ;
2411. };
2412. }FW\_MATCH\_VALUE;
2413. typedef enum \_tag\_FW\_MATCH\_TYPE
2414. {
2415. FW\_MATCH\_TYPE\_TRAFFIC\_MATCH,
2416. FW\_MATCH\_TYPE\_EQUAL,
2417. FW\_MATCH\_TYPE\_MAX
2418. }FW\_MATCH\_TYPE;
2419. typedef struct \_tag\_FW\_QUERY\_CONDITION
2420. {
2421. FW\_MATCH\_KEY matchKey;
2422. FW\_MATCH\_TYPE matchType;
2423. FW\_MATCH\_VALUE matchValue;
2424. }FW\_QUERY\_CONDITION, \*PFW\_QUERY\_CONDITION;
2425. typedef struct \_tag\_FW\_QUERY\_CONDITIONS
2426. {
2427. DWORD dwNumEntries;
2428. [size\_is(dwNumEntries)]
2429. FW\_QUERY\_CONDITION \*AndedConditions;
2430. }FW\_QUERY\_CONDITIONS, \*PFW\_QUERY\_CONDITIONS;
2431. typedef struct \_tag\_FW\_QUERY
2432. {
2433. WORD wSchemaVersion;
2434. UINT32 dwNumEntries;
2435. [size\_is(dwNumEntries)]
2436. FW\_QUERY\_CONDITIONS \*ORConditions;
2437. FW\_RULE\_STATUS Status;
2438. }FW\_QUERY, \*PFW\_QUERY;
2439. cpp\_quote("#endif //\_\_FIREWALL\_H\_")
2440. cpp\_quote("#define MIDL\_user\_allocate MIDL\_fw\_allocate")
2441. cpp\_quote("#define MIDL\_user\_free MIDL\_fw\_free")
2442. cpp\_quote("void \* \_\_RPC\_USER MIDL\_fw\_allocate(size\_t size);")
2443. cpp\_quote("void \_\_RPC\_USER MIDL\_fw\_free(void \* );")
2444. [
2445. uuid(6b5bdd1e-528c-422c-af8c-a4079be4fe48),
2446. version(1.0),
2447. pointer\_default(unique)
2448. ]
2449. interface RemoteFW
2450. {
2451. typedef
2452. handle\_t FW\_CONN\_HANDLE;
2453. typedef
2454. [context\_handle]
2455. HANDLE FW\_POLICY\_STORE\_HANDLE;
2456. typedef
2457. [ref]
2458. FW\_POLICY\_STORE\_HANDLE \*PFW\_POLICY\_STORE\_HANDLE;
2459. typedef
2460. [context\_handle]
2461. void\* FW\_PRODUCT\_HANDLE;
2462. DWORD
2463. RRPC\_FWOpenPolicyStore(
2464. [in] FW\_CONN\_HANDLE rpcConnHandle,
2465. [in] WORD BinaryVersion,
2466. [in, range(FW\_STORE\_TYPE\_INVALID+1, FW\_STORE\_TYPE\_MAX-1)] FW\_STORE\_TYPE StoreType,
2467. [in, range(FW\_POLICY\_ACCESS\_RIGHT\_INVALID+1, FW\_POLICY\_ACCESS\_RIGHT\_MAX-1)] FW\_POLICY\_ACCESS\_RIGHT AccessRight,
2468. [in] DWORD dwFlags,
2469. [out] PFW\_POLICY\_STORE\_HANDLE phPolicyStore
2470. );
2471. DWORD
2472. RRPC\_FWClosePolicyStore(
2473. [in] FW\_CONN\_HANDLE rpcConnHandle,
2474. [in, out] PFW\_POLICY\_STORE\_HANDLE phPolicyStore
2475. );
2476. DWORD
2477. RRPC\_FWRestoreDefaults([in] FW\_CONN\_HANDLE rpcConnHandle);
2478. DWORD
2479. RRPC\_FWGetGlobalConfig(
2480. [in] FW\_CONN\_HANDLE rpcConnHandle,
2481. [in] WORD BinaryVersion,
2482. [in] FW\_STORE\_TYPE StoreType,
2483. [in, range(FW\_GLOBAL\_CONFIG\_INVALID+1, FW\_GLOBAL\_CONFIG\_MAX-1)]
2484. FW\_GLOBAL\_CONFIG configID,
2485. [in] DWORD dwFlags, // Bit-wise combination of flags from FW\_CONFIG\_FLAGS
2486. [in, out, unique, size\_is(cbData), length\_is(\*pcbTransmittedLen)]
2487. BYTE\* pBuffer,
2488. [in] DWORD cbData,
2489. [in,out] LPDWORD pcbTransmittedLen,
2490. [out] LPDWORD pcbRequired
2491. );
2492. DWORD
2493. RRPC\_FWSetGlobalConfig(
2494. [in] FW\_CONN\_HANDLE rpcConnHandle,
2495. [in] WORD BinaryVersion,
2496. [in] FW\_STORE\_TYPE StoreType,
2497. [in, range(FW\_GLOBAL\_CONFIG\_INVALID+1, FW\_GLOBAL\_CONFIG\_MAX-1)] FW\_GLOBAL\_CONFIG configID,
2498. [in, unique, size\_is(dwBufSize)]
2499. BYTE \* lpBuffer,
2500. [in, range(0, 10\*1024)] DWORD dwBufSize
2501. );
2502. DWORD
2503. RRPC\_FWAddFirewallRule(
2504. [in] FW\_CONN\_HANDLE rpcConnHandle,
2505. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2506. [in] PFW\_RULE2\_0 pRule
2507. );
2508. DWORD
2509. RRPC\_FWSetFirewallRule(
2510. [in] FW\_CONN\_HANDLE rpcConnHandle,
2511. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2512. [in] PFW\_RULE2\_0 pRule
2513. );
2514. DWORD
2515. RRPC\_FWDeleteFirewallRule(
2516. [in] FW\_CONN\_HANDLE rpcConnHandle,
2517. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2518. [in, string, ref] LPCWSTR wszRuleID
2519. );
2520. DWORD
2521. RRPC\_FWDeleteAllFirewallRules(
2522. [in] FW\_CONN\_HANDLE rpcConnHandle,
2523. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore
2524. );
2525. DWORD
2526. RRPC\_FWEnumFirewallRules(
2527. [in] FW\_CONN\_HANDLE rpcConnHandle,
2528. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2529. [in] DWORD dwFilteredByStatus,
2530. [in] DWORD dwProfileFilter, // Bit-flags from FW\_PROFILE\_TYPE
2531. [in] WORD wFlags, // Bit-flags from ENUM\_RULES\_FLAGS
2532. [out, ref] DWORD \*pdwNumRules,
2533. [out] PFW\_RULE2\_0 \*ppRules
2534. );
2535. DWORD
2536. RRPC\_FWGetConfig(
2537. [in] FW\_CONN\_HANDLE rpcConnHandle,
2538. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2539. [in, range(FW\_PROFILE\_CONFIG\_ENABLE\_FW, FW\_PROFILE\_CONFIG\_MAX-1)] FW\_PROFILE\_CONFIG configID,
2540. [in] FW\_PROFILE\_TYPE Profile,
2541. [in] DWORD dwFlags, // Bit-wise combination of flags from FW\_CONFIG\_FLAGS
2542. [in, out, unique, size\_is(cbData), length\_is(\*pcbTransmittedLen)]
2543. BYTE\* pBuffer,
2544. [in] DWORD cbData,
2545. [in,out] LPDWORD pcbTransmittedLen,
2546. [out] LPDWORD pcbRequired
2547. );
2548. DWORD
2549. RRPC\_FWSetConfig(
2550. [in] FW\_CONN\_HANDLE rpcConnHandle,
2551. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2552. [in, range(FW\_PROFILE\_CONFIG\_ENABLE\_FW, FW\_PROFILE\_CONFIG\_MAX-1)] FW\_PROFILE\_CONFIG configID,
2553. [in] FW\_PROFILE\_TYPE Profile,
2554. [in, switch\_is(configID)] FW\_PROFILE\_CONFIG\_VALUE pConfig,
2555. [in, range(0, 10\*1024)] DWORD dwBufSize
2556. );
2557. DWORD
2558. RRPC\_FWAddConnectionSecurityRule(
2559. [in] FW\_CONN\_HANDLE rpcConnHandle,
2560. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2561. [in] PFW\_CS\_RULE2\_0 pRule
2562. );
2563. DWORD
2564. RRPC\_FWSetConnectionSecurityRule(
2565. [in] FW\_CONN\_HANDLE rpcConnHandle,
2566. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2567. [in] PFW\_CS\_RULE2\_0 pRule
2568. );
2569. DWORD
2570. RRPC\_FWDeleteConnectionSecurityRule(
2571. [in] FW\_CONN\_HANDLE rpcConnHandle,
2572. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2573. [in, string, ref] LPWSTR pRuleId
2574. );
2575. DWORD
2576. RRPC\_FWDeleteAllConnectionSecurityRules(
2577. [in] FW\_CONN\_HANDLE rpcConnHandle,
2578. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore
2579. );
2580. DWORD
2581. RRPC\_FWEnumConnectionSecurityRules(
2582. [in] FW\_CONN\_HANDLE rpcConnHandle,
2583. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2584. [in] DWORD dwFilteredByStatus,// Bit-flags from FW\_RULE\_STATUS\_CLASS
2585. [in] DWORD dwProfileFilter, // Bit-flags from FW\_PROFILE\_TYPE
2586. [in] WORD wFlags, // Bit-flags from FW\_ENUM\_RULES\_FLAGS
2587. [out, ref] DWORD \* pdwNumRules,
2588. [out] PFW\_CS\_RULE2\_0\* ppRules
2589. );
2590. DWORD
2591. RRPC\_FWAddAuthenticationSet(
2592. [in] FW\_CONN\_HANDLE rpcConnHandle,
2593. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2594. [in] PFW\_AUTH\_SET2\_10 pAuth
2595. );
2596. DWORD
2597. RRPC\_FWSetAuthenticationSet(
2598. [in] FW\_CONN\_HANDLE rpcConnHandle,
2599. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2600. [in] PFW\_AUTH\_SET2\_10 pAuth
2601. );
2602. DWORD
2603. RRPC\_FWDeleteAuthenticationSet(
2604. [in] FW\_CONN\_HANDLE rpcConnHandle,
2605. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2606. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)] FW\_IPSEC\_PHASE IpSecPhase,
2607. [in, string, ref] LPCWSTR wszSetId
2608. );
2609. DWORD
2610. RRPC\_FWDeleteAllAuthenticationSets(
2611. [in] FW\_CONN\_HANDLE rpcConnHandle,
2612. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2613. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)] FW\_IPSEC\_PHASE IpSecPhase
2614. );
2615. DWORD
2616. RRPC\_FWEnumAuthenticationSets(
2617. [in] FW\_CONN\_HANDLE rpcConnHandle,
2618. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2619. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)] FW\_IPSEC\_PHASE IpSecPhase,
2620. [in] DWORD dwFilteredByStatus,// Bit-flags from FW\_RULE\_STATUS\_CLASS
2621. [in] WORD wFlags, // Bit-flags from FW\_ENUM\_RULES\_FLAGS
2622. [out] DWORD\* pdwNumAuthSets,
2623. [out] PFW\_AUTH\_SET2\_10\* ppAuth
2624. );
2625. DWORD
2626. RRPC\_FWAddCryptoSet(
2627. [in] FW\_CONN\_HANDLE rpcConnHandle,
2628. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2629. [in] PFW\_CRYPTO\_SET pCrypto
2630. );
2631. DWORD
2632. RRPC\_FWSetCryptoSet(
2633. [in] FW\_CONN\_HANDLE rpcConnHandle,
2634. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2635. [in] PFW\_CRYPTO\_SET pCrypto
2636. );
2637. DWORD
2638. RRPC\_FWDeleteCryptoSet(
2639. [in] FW\_CONN\_HANDLE rpcConnHandle,
2640. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2641. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)] FW\_IPSEC\_PHASE IpSecPhase,
2642. [in, string, ref] LPCWSTR wszSetId
2643. );
2644. DWORD
2645. RRPC\_FWDeleteAllCryptoSets(
2646. [in] FW\_CONN\_HANDLE rpcConnHandle,
2647. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2648. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)] FW\_IPSEC\_PHASE IpSecPhase
2649. );
2650. DWORD
2651. RRPC\_FWEnumCryptoSets(
2652. [in] FW\_CONN\_HANDLE rpcConnHandle,
2653. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2654. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)] FW\_IPSEC\_PHASE IpSecPhase,
2655. [in] DWORD dwFilteredByStatus,// Bit-flags from FW\_RULE\_STATUS\_CLASS
2656. [in] WORD wFlags, // Bit-flags from FW\_ENUM\_RULES\_FLAGS
2657. [out, ref] DWORD\* pdwNumSets,
2658. [out] PFW\_CRYPTO\_SET\* ppCryptoSets
2659. );
2660. DWORD
2661. RRPC\_FWEnumPhase1SAs(
2662. [in] FW\_CONN\_HANDLE rpcConnHandle,
2663. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2664. [in,unique] PFW\_ENDPOINTS pEndpoints, //NULL or empty implies all endpoints
2665. [out,ref] DWORD\* pdwNumSAs,
2666. [out,size\_is( , \*pdwNumSAs)] PFW\_PHASE1\_SA\_DETAILS\* ppSAs
2667. );
2668. DWORD
2669. RRPC\_FWEnumPhase2SAs(
2670. [in] FW\_CONN\_HANDLE rpcConnHandle,
2671. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2672. [in, unique] PFW\_ENDPOINTS pEndpoints, //NULL or empty implies all endpoints
2673. [out,ref] DWORD\* pdwNumSAs,
2674. [out,size\_is( , \*pdwNumSAs)] PFW\_PHASE2\_SA\_DETAILS\* ppSAs
2675. );
2676. DWORD
2677. RRPC\_FWDeletePhase1SAs(
2678. [in] FW\_CONN\_HANDLE rpcConnHandle,
2679. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2680. [in, unique]PFW\_ENDPOINTS pEndpoints //NULL or empty implies all endpoints
2681. );
2682. DWORD
2683. RRPC\_FWDeletePhase2SAs(
2684. [in] FW\_CONN\_HANDLE rpcConnHandle,
2685. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2686. [in, unique]PFW\_ENDPOINTS pEndpoints //NULL or empty implies all endpoints
2687. );
2688. DWORD
2689. RRPC\_FWEnumProducts(
2690. [in] FW\_CONN\_HANDLE rpcConnHandle,
2691. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2692. [out] DWORD\* pdwNumProducts,
2693. [out, size\_is(,\*pdwNumProducts)] PFW\_PRODUCT\* ppProducts
2694. );
2695. DWORD
2696. RRPC\_FWAddMainModeRule(
2697. [in] FW\_CONN\_HANDLE rpcConnHandle,
2698. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2699. [in] PFW\_MM\_RULE pMMRule,
2700. [out] FW\_RULE\_STATUS \* pStatus
2701. );
2702. DWORD
2703. RRPC\_FWSetMainModeRule(
2704. [in] FW\_CONN\_HANDLE rpcConnHandle,
2705. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2706. [in] PFW\_MM\_RULE pMMRule,
2707. [out] FW\_RULE\_STATUS \* pStatus
2708. );
2709. DWORD
2710. RRPC\_FWDeleteMainModeRule(
2711. [in] FW\_CONN\_HANDLE rpcConnHandle,
2712. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2713. [in, string, ref] LPWSTR pRuleId
2714. );
2715. DWORD
2716. RRPC\_FWDeleteAllMainModeRules(
2717. [in] FW\_CONN\_HANDLE rpcConnHandle,
2718. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore
2719. );
2720. DWORD
2721. RRPC\_FWEnumMainModeRules(
2722. [in] FW\_CONN\_HANDLE rpcConnHandle,
2723. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2724. [in] DWORD dwFilteredByStatus,// Bit-flags from FW\_RULE\_STATUS\_CLASS
2725. [in] DWORD dwProfileFilter, // Bit-flags from FW\_PROFILE\_TYPE
2726. [in] WORD wFlags, // Bit-flags from FW\_ENUM\_RULES\_FLAGS
2727. [out, ref] DWORD\* pdwNumRules,
2728. [out] PFW\_MM\_RULE \* ppMMRules
2729. );
2730. DWORD
2731. RRPC\_FWQueryFirewallRules(
2732. [in] FW\_CONN\_HANDLE rpcConnHandle,
2733. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2734. [in] PFW\_QUERY pQuery, // Query selecting the rules to return
2735. [in] WORD wFlags, // Bit-flags from FW\_ENUM\_RULES\_FLAGS
2736. [out, ref] DWORD\* pdwNumRules,
2737. [out] PFW\_RULE2\_10 \* ppRules
2738. );
2739. DWORD
2740. RRPC\_FWQueryConnectionSecurityRules2\_10(
2741. [in] FW\_CONN\_HANDLE rpcConnHandle,
2742. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2743. [in] PFW\_QUERY pQuery, // Query selecting the rules to return
2744. [in] WORD wFlags, // Bit-flags from FW\_ENUM\_RULES\_FLAGS
2745. [out, ref] DWORD\* pdwNumRules,
2746. [out] PFW\_CS\_RULE2\_10 \* ppRules
2747. );
2748. DWORD
2749. RRPC\_FWQueryMainModeRules(
2750. [in] FW\_CONN\_HANDLE rpcConnHandle,
2751. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2752. [in] PFW\_QUERY pQuery, // Query selecting the rules to return
2753. [in] WORD wFlags, // Bit-flags from FW\_ENUM\_RULES\_FLAGS
2754. [out, ref] DWORD\* pdwNumRules,
2755. [out] PFW\_MM\_RULE \* ppMMRules
2756. );
2757. DWORD
2758. RRPC\_FWQueryAuthenticationSets(
2759. [in] FW\_CONN\_HANDLE rpcConnHandle,
2760. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2761. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)] FW\_IPSEC\_PHASE IPsecPhase,
2762. [in] PFW\_QUERY pQuery, // Query selecting the rules to return
2763. [in] WORD wFlags, // Bit-flags from FW\_ENUM\_RULES\_FLAGS
2764. [out, ref] DWORD\* pdwNumSets,
2765. [out] PFW\_AUTH\_SET2\_10 \* ppAuthSets
2766. );
2767. DWORD
2768. RRPC\_FWQueryCryptoSets(
2769. [in] FW\_CONN\_HANDLE rpcConnHandle,
2770. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2771. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)] FW\_IPSEC\_PHASE IPsecPhase,
2772. [in] PFW\_QUERY pQuery, // Query selecting the rules to return
2773. [in] WORD wFlags, // Bit-flags from FW\_ENUM\_RULES\_FLAGS
2774. [out, ref] DWORD\* pdwNumSets,
2775. [out] PFW\_CRYPTO\_SET \* ppCryptoSets
2776. );
2777. DWORD
2778. RRPC\_FWEnumNetworks(
2779. [in] FW\_CONN\_HANDLE rpcConnHandle,
2780. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2781. [out] DWORD\* pdwNumNetworks,
2782. [out, size\_is(,\*pdwNumNetworks)] PFW\_NETWORK\* ppNetworks
2783. );
2784. DWORD
2785. RRPC\_FWEnumAdapters(
2786. [in] FW\_CONN\_HANDLE rpcConnHandle,
2787. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2788. [out] DWORD\* pdwNumAdapters,
2789. [out, size\_is(,\*pdwNumAdapters)] PFW\_ADAPTER\* ppAdapters
2790. );
2791. DWORD
2792. RRPC\_FWGetGlobalConfig2\_10(
2793. [in] FW\_CONN\_HANDLE rpcConnHandle,
2794. [in] WORD BinaryVersion,
2795. [in] FW\_STORE\_TYPE StoreType,
2796. [in, range(FW\_GLOBAL\_CONFIG\_INVALID+1, FW\_GLOBAL\_CONFIG\_MAX-1)]
2797. FW\_GLOBAL\_CONFIG configID,
2798. [in] DWORD dwFlags, // Bit-wise combination of flags from FW\_CONFIG\_FLAGS
2799. [in, out, unique, size\_is(cbData), length\_is(\*pcbTransmittedLen)]
2800. BYTE\* pBuffer,
2801. [in] DWORD cbData,
2802. [in,out] LPDWORD pcbTransmittedLen,
2803. [out] LPDWORD pcbRequired,
2804. [out] FW\_RULE\_ORIGIN\_TYPE \* pOrigin
2805. );
2806. DWORD
2807. RRPC\_FWGetConfig2\_10(
2808. [in] FW\_CONN\_HANDLE rpcConnHandle,
2809. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2810. [in, range(FW\_PROFILE\_CONFIG\_ENABLE\_FW, FW\_PROFILE\_CONFIG\_MAX-1)] FW\_PROFILE\_CONFIG configID,
2811. [in] FW\_PROFILE\_TYPE Profile,
2812. [in] DWORD dwFlags, // Bit-wise combination of flags from FW\_CONFIG\_FLAGS
2813. [in, out, unique, size\_is(cbData), length\_is(\*pcbTransmittedLen)]
2814. BYTE\* pBuffer,
2815. [in] DWORD cbData,
2816. [in,out] LPDWORD pcbTransmittedLen,
2817. [out] LPDWORD pcbRequired,
2818. [out] FW\_RULE\_ORIGIN\_TYPE \* pOrigin
2819. );
2820. DWORD
2821. RRPC\_FWAddFirewallRule2\_10(
2822. [in] FW\_CONN\_HANDLE rpcConnHandle,
2823. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2824. [in] PFW\_RULE2\_10 pRule,
2825. [out] FW\_RULE\_STATUS \* pStatus
2826. );
2827. DWORD
2828. RRPC\_FWSetFirewallRule2\_10(
2829. [in] FW\_CONN\_HANDLE rpcConnHandle,
2830. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2831. [in] PFW\_RULE2\_10 pRule,
2832. [out] FW\_RULE\_STATUS \* pStatus
2833. );
2834. DWORD
2835. RRPC\_FWEnumFirewallRules2\_10(
2836. [in] FW\_CONN\_HANDLE rpcConnHandle,
2837. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2838. [in] DWORD dwFilteredByStatus,
2839. [in] DWORD dwProfileFilter, // Bit-flags from FW\_PROFILE\_TYPE
2840. [in] WORD wFlags, // Bit-flags from ENUM\_RULES\_FLAGS
2841. [out, ref] DWORD \*pdwNumRules,
2842. [out] PFW\_RULE2\_10 \*ppRules
2843. );
2844. DWORD
2845. RRPC\_FWAddConnectionSecurityRule2\_10(
2846. [in] FW\_CONN\_HANDLE rpcConnHandle,
2847. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2848. [in] PFW\_CS\_RULE2\_10 pRule,
2849. [out] FW\_RULE\_STATUS \* pStatus
2850. );
2851. DWORD
2852. RRPC\_FWSetConnectionSecurityRule2\_10(
2853. [in] FW\_CONN\_HANDLE rpcConnHandle,
2854. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2855. [in] PFW\_CS\_RULE2\_10 pRule,
2856. [out] FW\_RULE\_STATUS \* pStatus
2857. );
2858. DWORD
2859. RRPC\_FWEnumConnectionSecurityRules2\_10(
2860. [in] FW\_CONN\_HANDLE rpcConnHandle,
2861. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2862. [in] DWORD dwFilteredByStatus,// Bit-flags from FW\_RULE\_STATUS\_CLASS
2863. [in] DWORD dwProfileFilter, // Bit-flags from FW\_PROFILE\_TYPE
2864. [in] WORD wFlags, // Bit-flags from FW\_ENUM\_RULES\_FLAGS
2865. [out, ref] DWORD \* pdwNumRules,
2866. [out] PFW\_CS\_RULE2\_10\* ppRules
2867. );
2868. DWORD
2869. RRPC\_FWAddAuthenticationSet2\_10(
2870. [in] FW\_CONN\_HANDLE rpcConnHandle,
2871. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2872. [in] PFW\_AUTH\_SET2\_10 pAuth,
2873. [out] FW\_RULE\_STATUS \* pStatus
2874. );
2875. DWORD
2876. RRPC\_FWSetAuthenticationSet2\_10(
2877. [in] FW\_CONN\_HANDLE rpcConnHandle,
2878. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2879. [in] PFW\_AUTH\_SET2\_10 pAuth,
2880. [out] FW\_RULE\_STATUS \* pStatus
2881. );
2882. DWORD
2883. RRPC\_FWEnumAuthenticationSets2\_10(
2884. [in] FW\_CONN\_HANDLE rpcConnHandle,
2885. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2886. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)] FW\_IPSEC\_PHASE IpSecPhase,
2887. [in] DWORD dwFilteredByStatus,// Bit-flags from FW\_RULE\_STATUS\_CLASS
2888. [in] WORD wFlags, // Bit-flags from FW\_ENUM\_RULES\_FLAGS
2889. [out] DWORD\* pdwNumAuthSets,
2890. [out] PFW\_AUTH\_SET2\_10\* ppAuth
2891. );
2892. DWORD
2893. RRPC\_FWAddCryptoSet2\_10(
2894. [in] FW\_CONN\_HANDLE rpcConnHandle,
2895. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2896. [in] PFW\_CRYPTO\_SET pCrypto,
2897. [out] FW\_RULE\_STATUS \* pStatus
2898. );
2899. DWORD
2900. RRPC\_FWSetCryptoSet2\_10(
2901. [in] FW\_CONN\_HANDLE rpcConnHandle,
2902. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2903. [in] PFW\_CRYPTO\_SET pCrypto,
2904. [out] FW\_RULE\_STATUS \* pStatus
2905. );
2906. DWORD
2907. RRPC\_FWEnumCryptoSets2\_10(
2908. [in] FW\_CONN\_HANDLE rpcConnHandle,
2909. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2910. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)] FW\_IPSEC\_PHASE IpSecPhase,
2911. [in] DWORD dwFilteredByStatus,// Bit-flags from FW\_RULE\_STATUS\_CLASS
2912. [in] WORD wFlags, // Bit-flags from FW\_ENUM\_RULES\_FLAGS
2913. [out, ref] DWORD\* pdwNumSets,
2914. [out] PFW\_CRYPTO\_SET\* ppCryptoSets
2915. );
2916. DWORD
2917. RRPC\_FWAddConnectionSecurityRule2\_20(
2918. [in] FW\_CONN\_HANDLE rpcConnHandle,
2919. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2920. [in] PFW\_CS\_RULE pRule,
2921. [out] FW\_RULE\_STATUS \* pStatus
2922. );
2923. DWORD
2924. RRPC\_FWSetConnectionSecurityRule2\_20(
2925. [in] FW\_CONN\_HANDLE rpcConnHandle,
2926. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2927. [in] PFW\_CS\_RULE pRule,
2928. [out] FW\_RULE\_STATUS \* pStatus
2929. );
2930. DWORD
2931. RRPC\_FWEnumConnectionSecurityRules2\_20(
2932. [in] FW\_CONN\_HANDLE rpcConnHandle,
2933. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2934. [in] DWORD dwFilteredByStatus,// Bit-flags from FW\_RULE\_STATUS\_CLASS
2935. [in] DWORD dwProfileFilter, // Bit-flags from FW\_PROFILE\_TYPE
2936. [in] WORD wFlags, // Bit-flags from FW\_ENUM\_RULES\_FLAGS
2937. [out, ref] DWORD \* pdwNumRules,
2938. [out] PFW\_CS\_RULE\* ppRules
2939. );
2940. DWORD
2941. RRPC\_FWQueryConnectionSecurityRules2\_20(
2942. [in] FW\_CONN\_HANDLE rpcConnHandle,
2943. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2944. [in] PFW\_QUERY pQuery, // Query selecting the rules to return
2945. [in] WORD wFlags, // Bit-flags from FW\_ENUM\_RULES\_FLAGS
2946. [out, ref] DWORD\* pdwNumRules,
2947. [out] PFW\_CS\_RULE \* ppRules
2948. );
2949. DWORD
2950. RRPC\_FWAddAuthenticationSet2\_20(
2951. [in] FW\_CONN\_HANDLE rpcConnHandle,
2952. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2953. [in] PFW\_AUTH\_SET pAuth,
2954. [out] FW\_RULE\_STATUS \* pStatus
2955. );
2956. DWORD
2957. RRPC\_FWSetAuthenticationSet2\_20(
2958. [in] FW\_CONN\_HANDLE rpcConnHandle,
2959. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2960. [in] PFW\_AUTH\_SET pAuth,
2961. [out] FW\_RULE\_STATUS \* pStatus
2962. );
2963. DWORD
2964. RRPC\_FWEnumAuthenticationSets2\_20(
2965. [in] FW\_CONN\_HANDLE rpcConnHandle,
2966. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2967. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)] FW\_IPSEC\_PHASE IpSecPhase,
2968. [in] DWORD dwFilteredByStatus,// Bit-flags from FW\_RULE\_STATUS\_CLASS
2969. [in] WORD wFlags, // Bit-flags from FW\_ENUM\_RULES\_FLAGS
2970. [out] DWORD\* pdwNumAuthSets,
2971. [out] PFW\_AUTH\_SET\* ppAuth
2972. );
2973. DWORD
2974. RRPC\_FWQueryAuthenticationSets2\_20(
2975. [in] FW\_CONN\_HANDLE rpcConnHandle,
2976. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2977. [in, range(FW\_IPSEC\_PHASE\_INVALID+1, FW\_IPSEC\_PHASE\_MAX-1)] FW\_IPSEC\_PHASE IPsecPhase,
2978. [in] PFW\_QUERY pQuery, // Query selecting the rules to return
2979. [in] WORD wFlags, // Bit-flags from FW\_ENUM\_RULES\_FLAGS
2980. [out, ref] DWORD\* pdwNumSets,
2981. [out] PFW\_AUTH\_SET\* ppAuthSets
2982. );
2983. DWORD
2984. RRPC\_FWAddFirewallRule2\_20(
2985. [in] FW\_CONN\_HANDLE rpcConnHandle,
2986. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2987. [in] PFW\_RULE2\_20 pRule,
2988. [out] FW\_RULE\_STATUS \* pStatus
2989. );
2990. DWORD
2991. RRPC\_FWSetFirewallRule2\_20(
2992. [in] FW\_CONN\_HANDLE rpcConnHandle,
2993. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
2994. [in] PFW\_RULE2\_20 pRule,
2995. [out] FW\_RULE\_STATUS \* pStatus
2996. );
2997. DWORD
2998. RRPC\_FWEnumFirewallRules2\_20(
2999. [in] FW\_CONN\_HANDLE rpcConnHandle,
3000. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
3001. [in] DWORD dwFilteredByStatus,
3002. [in] DWORD dwProfileFilter, // Bit-flags from FW\_PROFILE\_TYPE
3003. [in] WORD wFlags, // Bit-flags from ENUM\_RULES\_FLAGS
3004. [out, ref] DWORD \*pdwNumRules,
3005. [out] PFW\_RULE2\_20 \*ppRules
3006. );
3007. DWORD
3008. RRPC\_FWQueryFirewallRules2\_20(
3009. [in] FW\_CONN\_HANDLE rpcConnHandle,
3010. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
3011. [in] PFW\_QUERY pQuery, // Query selecting the rules to return
3012. [in] WORD wFlags, // Bit-flags from FW\_ENUM\_RULES\_FLAGS
3013. [out, ref] DWORD\* pdwNumRules,
3014. [out] PFW\_RULE2\_20 \* ppRules
3015. );
3016. DWORD
3017. RRPC\_FWAddFirewallRule2\_24(
3018. [in] FW\_CONN\_HANDLE rpcConnHandle,
3019. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
3020. [in] PFW\_RULE2\_24 pRule,
3021. [out] FW\_RULE\_STATUS \* pStatus
3022. );
3023. DWORD
3024. RRPC\_FWSetFirewallRule2\_24(
3025. [in] FW\_CONN\_HANDLE rpcConnHandle,
3026. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
3027. [in] PFW\_RULE2\_24 pRule,
3028. [out] FW\_RULE\_STATUS \* pStatus
3029. );
3030. DWORD
3031. RRPC\_FWEnumFirewallRules2\_24(
3032. [in] FW\_CONN\_HANDLE rpcConnHandle,
3033. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
3034. [in] DWORD dwFilteredByStatus,
3035. [in] DWORD dwProfileFilter, // Bit-flags from FW\_PROFILE\_TYPE
3036. [in] WORD wFlags, // Bit-flags from ENUM\_RULES\_FLAGS
3037. [out, ref] DWORD \*pdwNumRules,
3038. [out] PFW\_RULE2\_24 \*ppRules
3039. );
3040. DWORD
3041. RRPC\_FWQueryFirewallRules2\_24(
3042. [in] FW\_CONN\_HANDLE rpcConnHandle,
3043. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
3044. [in] PFW\_QUERY pQuery, // Query selecting the rules to return
3045. [in] WORD wFlags, // Bit-flags from FW\_ENUM\_RULES\_FLAGS
3046. [out, ref] DWORD\* pdwNumRules,
3047. [out] PFW\_RULE2\_24 \*ppRules
3048. );
3049. DWORD
3050. RRPC\_FWAddFirewallRule2\_25(
3051. [in] FW\_CONN\_HANDLE rpcConnHandle,
3052. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
3053. [in] PFW\_RULE2\_25 pRule,
3054. [out] FW\_RULE\_STATUS \* pStatus
3055. );
3056. DWORD
3057. RRPC\_FWSetFirewallRule2\_25(
3058. [in] FW\_CONN\_HANDLE rpcConnHandle,
3059. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
3060. [in] PFW\_RULE2\_25 pRule,
3061. [out] FW\_RULE\_STATUS \* pStatus
3062. );
3063. DWORD
3064. RRPC\_FWEnumFirewallRules2\_25(
3065. [in] FW\_CONN\_HANDLE rpcConnHandle,
3066. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
3067. [in] DWORD dwFilteredByStatus,
3068. [in] DWORD dwProfileFilter, // Bit-flags from FW\_PROFILE\_TYPE
3069. [in] WORD wFlags, // Bit-flags from ENUM\_RULES\_FLAGS
3070. [out, ref] DWORD \*pdwNumRules,
3071. [out] PFW\_RULE2\_25 \*ppRules
3072. );
3073. DWORD
3074. RRPC\_FWQueryFirewallRules2\_25(
3075. [in] FW\_CONN\_HANDLE rpcConnHandle,
3076. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
3077. [in] PFW\_QUERY pQuery, // Query selecting the rules to return
3078. [in] WORD wFlags, // Bit-flags from FW\_ENUM\_RULES\_FLAGS
3079. [out, ref] DWORD\* pdwNumRules,
3080. [out] PFW\_RULE2\_25 \* ppRules
3081. );
3082. DWORD
3083. RRPC\_FWAddFirewallRule2\_26(
3084. [in] FW\_CONN\_HANDLE rpcConnHandle,
3085. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
3086. [in] PFW\_RULE2\_26 pRule,
3087. [out] FW\_RULE\_STATUS \* pStatus
3088. );
3089. DWORD
3090. RRPC\_FWSetFirewallRule2\_26(
3091. [in] FW\_CONN\_HANDLE rpcConnHandle,
3092. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
3093. [in] PFW\_RULE2\_26 pRule,
3094. [out] FW\_RULE\_STATUS \* pStatus
3095. );
3096. DWORD
3097. RRPC\_FWEnumFirewallRules2\_26(
3098. [in] FW\_CONN\_HANDLE rpcConnHandle,
3099. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
3100. [in] DWORD dwFilteredByStatus,
3101. [in] DWORD dwProfileFilter, // Bit-flags from FW\_PROFILE\_TYPE
3102. [in] WORD wFlags, // Bit-flags from ENUM\_RULES\_FLAGS
3103. [out, ref] DWORD \*pdwNumRules,
3104. [out] PFW\_RULE2\_26 \*ppRules
3105. );
3106. DWORD
3107. RRPC\_FWQueryFirewallRules2\_26(
3108. [in] FW\_CONN\_HANDLE rpcConnHandle,
3109. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
3110. [in] PFW\_QUERY pQuery, // Query selecting the rules to return
3111. [in] WORD wFlags, // Bit-flags from FW\_ENUM\_RULES\_FLAGS
3112. [out, ref] DWORD\* pdwNumRules,
3113. [out] PFW\_RULE2\_26 \* ppRules
3114. );
3115. DWORD
3116. RRPC\_FWAddFirewallRule2\_27(
3117. [in] FW\_CONN\_HANDLE rpcConnHandle,
3118. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
3119. [in] PFW\_RULE pRule,
3120. [out] FW\_RULE\_STATUS \* pStatus
3121. );
3122. DWORD
3123. RRPC\_FWSetFirewallRule2\_27(
3124. [in] FW\_CONN\_HANDLE rpcConnHandle,
3125. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
3126. [in] PFW\_RULE pRule,
3127. [out] FW\_RULE\_STATUS \* pStatus
3128. );
3129. DWORD
3130. RRPC\_FWEnumFirewallRules2\_27(
3131. [in] FW\_CONN\_HANDLE rpcConnHandle,
3132. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
3133. [in] DWORD dwFilteredByStatus,
3134. [in] DWORD dwProfileFilter, // Bit-flags from FW\_PROFILE\_TYPE
3135. [in] WORD wFlags, // Bit-flags from ENUM\_RULES\_FLAGS
3136. [out, ref] DWORD \*pdwNumRules,
3137. [out] PFW\_RULE \*ppRules
3138. );
3139. DWORD
3140. RRPC\_FWQueryFirewallRules2\_27(
3141. [in] FW\_CONN\_HANDLE rpcConnHandle,
3142. [in] FW\_POLICY\_STORE\_HANDLE hPolicyStore,
3143. [in] PFW\_QUERY pQuery, // Query selecting the rules to return
3144. [in] WORD wFlags, // Bit-flags from FW\_ENUM\_RULES\_FLAGS
3145. [out, ref] DWORD\* pdwNumRules,
3146. [out] PFW\_RULE \* ppRules
3147. );
3148. }

# 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.

The following tables show the relationships between Microsoft product versions or supplemental software and the roles they perform.

| Windows Client Releases | Server Role | Client Role |
| --- | --- | --- |
| Windows Vista operating system | Yes | Yes |
| Windows 7 operating system | Yes | Yes |
| Windows 8 operating system | Yes | Yes |
| Windows 8.1 operating system | Yes | Yes |
| Windows 10 operating system | Yes | Yes |

| Windows Server Releases | Server Role | Client Role |
| --- | --- | --- |
| Windows Server 2008 operating system | Yes | Yes |
| Windows Server 2008 R2 operating system | Yes | Yes |
| Windows Server 2012 operating system | Yes | Yes |
| Windows Server 2012 R2 operating system | Yes | Yes |
| Windows Server 2016 operating system | Yes | Yes |
| Windows Server operating system | Yes | Yes |

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 1.7](#Appendix_A_Target_1): Policy versions are mapped to Windows releases as follows:

| Windows Client Release | Windows Server Release | Policy Version |
| --- | --- | --- |
| Windows Vista |  | 0x0200 |
| Windows Vista operating system with Service Pack 1 (SP1) | Windows Server 2008 | 0x0201 |
| Windows 7 | Windows Server 2008 R2 operating system | 0x020A |
| Windows 8 | Windows Server 2012 | 0x0214 |
| Windows 8.1 | Windows Server 2012 R2 | 0x0216 |
| Windows 10 |  | 0x0218, 0x0219 |
| Windows 10 v1607 operating system | Windows Server 2016 | 0x021A |
| Windows 10 v1703 operating system |  | 0x021B |
| Windows 10 v1709 operating system | Windows Server operating system | Ox021B |

[<2> Section 2.2.6](#Appendix_A_Target_2): For Windows Vista SP1, Windows Server 2008, Windows 7, and Windows Server 2008 R2, unspecified addresses are allowed. Unspecified addresses are also allowed on Windows Vista if the Security Update for Windows Vista specified in [[MSKB-935807]](https://go.microsoft.com/fwlink/?LinkId=122599) is applied.

[<3> Section 2.2.31](#Appendix_A_Target_3): During server initialization, Windows uses default values to initialize the Phase 1 and Phase 2 primary **AuthenticationSet** objects if these objects are not already present in **LocalStore** and **GroupPolicyRSoPStore**. The same defaults are used for both **LocalStore** and **GroupPolicyRSoPStore**. These defaults are as follows:

1. #define FW\_DEFAULT\_P1\_PRIMARY\_AUTH\_SET\_NAME\_STR
2. L"Default Phase1 Primary AuthSet"
3. #define FW\_DEFAULT\_P2\_PRIMARY\_AUTH\_SET\_NAME\_STR
4. L"Default Phase2 Primary AuthSet"
5. #define RTL\_NUMBER\_OF(A) (sizeof(A)/sizeof((A)[0]))
6. FW\_AUTH\_SUITE g\_DefaultPrimaryAuthSuitePhase1[] =
7. {
8. { FW\_AUTH\_METHOD\_MACHINE\_KERB, {0} }
9. };
10. FW\_AUTH\_SET g\_DefaultPrimaryAuthSetPhase1 =
11. {
12. NULL,
13. 0x0200,
14. FW\_IPSEC\_PHASE\_1,
15. L"{E5A5D32A-4BCE-4e4d-B07F-4AB1BA7E5FE3}",
16. FW\_DEFAULT\_P1\_PRIMARY\_AUTH\_SET\_NAME\_STR,
17. FW\_DEFAULT\_P1\_PRIMARY\_AUTH\_SET\_NAME\_STR,
18. NULL,
19. RTL\_NUMBER\_OF(g\_DefaultPrimaryAuthSuitePhase1),
20. g\_DefaultPrimaryAuthSuitePhase1,
21. FW\_RULE\_ORIGIN\_HARDCODED,
22. NULL,
23. FW\_RULE\_STATUS\_OK,
24. 0
25. };
26. FW\_AUTH\_SET g\_DefaultPrimaryAuthSetPhase2 =
27. {
28. NULL,
29. 0x0200,
30. FW\_IPSEC\_PHASE\_2,
31. L"{E5A5D32A-4BCE-4e4d-B07F-4AB1BA7E5FE4}",
32. FW\_DEFAULT\_P2\_PRIMARY\_AUTH\_SET\_NAME\_STR,
33. FW\_DEFAULT\_P2\_PRIMARY\_AUTH\_SET\_NAME\_STR,
34. NULL,
35. 0,
36. NULL,
37. FW\_RULE\_ORIGIN\_HARDCODED,
38. NULL,
39. FW\_RULE\_STATUS\_OK,
40. 0
41. };

During server initialization, Windows uses default values to initialize the Phase 1 and Phase 2 primary **CryptoSet** objects if these objects are not already present in **LocalStore** or **GroupPolicyRSoPStore**. The same defaults are used for both **LocalStore** and **GroupPolicyRSoPStore**. These defaults are as follows:

1. #define FW\_DEFAULT\_P1\_PRIMARY\_CRYPTO\_SET\_NAME\_STR
2. L"Default Phase1 Primary CryptoSet"
3. #define FW\_DEFAULT\_P2\_PRIMARY\_CRYPTO\_SET\_NAME\_STR
4. L"Default Phase2 Primary CryptoSet"
5. FW\_PHASE1\_CRYPTO\_SUITE g\_DefaultPrimaryCryptoSuitesPhase1[] =
6. {
7. {FW\_CRYPTO\_KEY\_EXCHANGE\_DH2,
8. FW\_CRYPTO\_ENCRYPTION\_AES128,
9. FW\_CRYPTO\_HASH\_SHA1},
10. {FW\_CRYPTO\_KEY\_EXCHANGE\_DH2,
11. FW\_CRYPTO\_ENCRYPTION\_3DES,
12. FW\_CRYPTO\_HASH\_SHA1}
13. };
15. FW\_CRYPTO\_SET g\_DefaultPrimaryCryptoSetPhase1 =
16. {
17. NULL,
18. 0x0200,
19. FW\_IPSEC\_PHASE\_1,
20. L"{E5A5D32A-4BCE-4e4d-B07F-4AB1BA7E5FE1}",
21. FW\_DEFAULT\_P1\_PRIMARY\_CRYPTO\_SET\_NAME\_STR,
22. FW\_DEFAULT\_P1\_PRIMARY\_CRYPTO\_SET\_NAME\_STR,
23. NULL,
24. {
25. 0, // flags
26. 0, // RTL\_NUMBER\_OF(g\_DefaultPrimaryCryptoSuitesPhase1),
27. 0, // g\_DefaultPrimaryCryptoSuitesPhase1,
28. 0, //480,
29. 0
30. },
31. FW\_RULE\_ORIGIN\_HARDCODED,
32. NULL,
33. FW\_RULE\_STATUS\_OK,
34. 0
35. };
37. FW\_PHASE2\_CRYPTO\_SUITE g\_DefaultPrimaryCryptoSuitesPhase2[] =
38. {
39. {FW\_CRYPTO\_PROTOCOL\_ESP,
40. FW\_CRYPTO\_HASH\_NONE,
41. FW\_CRYPTO\_HASH\_SHA1,
42. FW\_CRYPTO\_ENCRYPTION\_NONE,
43. FW\_DEFAULT\_CRYPTO\_PHASE2\_TIMEOUT\_MINUTES,
44. FW\_DEFAULT\_CRYPTO\_PHASE2\_TIMEOUT\_KBYTES},
45. {FW\_CRYPTO\_PROTOCOL\_ESP,
46. FW\_CRYPTO\_HASH\_NONE,
47. FW\_CRYPTO\_HASH\_SHA1,
48. FW\_CRYPTO\_ENCRYPTION\_AES128,
49. FW\_DEFAULT\_CRYPTO\_PHASE2\_TIMEOUT\_MINUTES,
50. FW\_DEFAULT\_CRYPTO\_PHASE2\_TIMEOUT\_KBYTES},
51. {FW\_CRYPTO\_PROTOCOL\_ESP,
52. FW\_CRYPTO\_HASH\_NONE,
53. FW\_CRYPTO\_HASH\_SHA1,
54. FW\_CRYPTO\_ENCRYPTION\_3DES,
55. FW\_DEFAULT\_CRYPTO\_PHASE2\_TIMEOUT\_MINUTES,
56. FW\_DEFAULT\_CRYPTO\_PHASE2\_TIMEOUT\_KBYTES},
57. {FW\_CRYPTO\_PROTOCOL\_AH,
58. FW\_CRYPTO\_HASH\_SHA1,
59. FW\_CRYPTO\_HASH\_NONE,
60. FW\_CRYPTO\_ENCRYPTION\_NONE,
61. FW\_DEFAULT\_CRYPTO\_PHASE2\_TIMEOUT\_MINUTES,
62. FW\_DEFAULT\_CRYPTO\_PHASE2\_TIMEOUT\_KBYTES}
63. };
65. FW\_CRYPTO\_SET g\_DefaultPrimaryCryptoSetPhase2 =
66. {
67. NULL,
68. 0x0200,
69. FW\_IPSEC\_PHASE\_2,
70. L"{E5A5D32A-4BCE-4e4d-B07F-4AB1BA7E5FE2}",
71. FW\_DEFAULT\_P2\_PRIMARY\_CRYPTO\_SET\_NAME\_STR,
72. FW\_DEFAULT\_P2\_PRIMARY\_CRYPTO\_SET\_NAME\_STR,
73. NULL,
74. {
75. {
76. 0, // FW\_PHASE2\_CRYPTO\_PFS\_DISABLE,
77. 0, // RTL\_NUMBER\_OF(g\_DefaultPrimaryCryptoSuitesPhase2),
78. 0, // g\_DefaultPrimaryCryptoSuitesPhase2
79. }
80. },
81. FW\_RULE\_ORIGIN\_HARDCODED,
82. NULL,
83. FW\_RULE\_STATUS\_OK,
84. 0
85. };
86. void FwDefaultPrimaryCryptoSetsInit()
87. {
88. // Init Phase 1 Crypto.
89. g\_DefaultPrimaryCryptoSetPhase1.dwNumPhase1Suites =
90. RTL\_NUMBER\_OF(g\_DefaultPrimaryCryptoSuitesPhase1);
91. g\_DefaultPrimaryCryptoSetPhase1.pPhase1Suites =
92. g\_DefaultPrimaryCryptoSuitesPhase1;
93. g\_DefaultPrimaryCryptoSetPhase1.dwTimeOutMinutes = 480;
94. //Init Phase 2 Crypto
95. g\_DefaultPrimaryCryptoSetPhase2.Pfs =
96. FW\_PHASE2\_CRYPTO\_PFS\_DISABLE;
97. g\_DefaultPrimaryCryptoSetPhase2.dwNumPhase2Suites =
98. RTL\_NUMBER\_OF(g\_DefaultPrimaryCryptoSuitesPhase2);
99. g\_DefaultPrimaryCryptoSetPhase2.pPhase2Suites =
100. g\_DefaultPrimaryCryptoSuitesPhase2;
101. }

[<4> Section 2.2.36](#Appendix_A_Target_4): Windows uses the three fields of the [FW\_OS\_PLATFORM](#Section_aac2be8cf4ab4b0a995774462137fd8a) data type to identify Windows platform types. The fields in this data type correspond to the fields of the Windows OSVERSIONINFOEX data type (for more information, see [[MSDN-OSVERSIONINFOEX]](https://go.microsoft.com/fwlink/?LinkId=90057)). The **bPlatform** field in this specification corresponds to the **dwPlatformId** field in MSDN. The **bMajorVersion** field in this specification corresponds to the **dwMajorVersion** field in MSDN. The **bMinorVersion** field in this specification corresponds to the **dwMinorVersion** field in MSDN. The Windows firewall and advanced security components extract the OSVERSIONINFOEX values and use them to enforce PlatformValidityList conditions in [FW\_RULE (section 2.2.36)](#Section_8c008258166d46d49090f2ffaa01be4b) and [FW\_CS\_RULE (section 2.2.54)](#Section_0d0641105f2e4b68aa63032c6cd5e4c6) rules.

[<5> Section 2.2.36](#Appendix_A_Target_5): Rules with **wSchemaVersion** less than 0x000200 but greater than or equal to 0x000100 are not allowed to be written to the local store.

[<6> Section 2.2.36](#Appendix_A_Target_6): On Windows 7 and Windows Server 2008 R2 the **wszRuleId** size cannot be greater than or equal to 512 characters. On Windows Vista and Windows Server 2008 it cannot be greater than or equal to 1000 characters.

[<7> Section 2.2.37](#Appendix_A_Target_7): When Windows is operating in [**stealth mode**](#gt_60f8407f-3974-42e5-b542-9156dcd18b82), it blocks the following outbound packets:

* ICMP Destination Unreachable
* ICMP Parameter Problem for IPv6 only
* TCP Reset (RST) packets sent because no application is listening on the destination port

[<8> Section 2.2.37](#Appendix_A_Target_8): In Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, the FW\_PROFILE\_CONFIG\_LOG\_IGNORED\_RULES option is ignored.

[<9> Section 2.2.37](#Appendix_A_Target_9): When an application is blocked from listening on a port and inbound notifications are not disabled, Windows displays a notification to the user only when there is not an FW\_RULE object in the [**Group Policy**](#gt_defe8c22-1365-4e5e-abf7-46ad112d3bda) RSoP, local, or dynamic policy stores with a **wszLocalApplication** field that matches the application.

[<10> Section 2.2.42](#Appendix_A_Target_10): Windows selects a default value for the profile configuration options and the global configurations options. These configurations default values are secure, and it is recommended to use these values as default values. Profile configuration options default values:

1. FW\_PROFILE\_CONFIG\_ENABLE\_FW .- TRUE.
2. FW\_PROFILE\_CONFIG\_DISABLE\_STEALTH\_MODE .- FALSE.
3. FW\_PROFILE\_CONFIG\_SHIELDED .- FALSE.
4. FW\_PROFILE\_CONFIG\_DISABLE\_UNICAST\_RESPONSES\_TO\_MULTICAST\_BROADCAST
5. .- FALSE.
6. FW\_PROFILE\_CONFIG\_LOG\_DROPPED\_PACKETS .- FALSE.
7. FW\_PROFILE\_CONFIG\_LOG\_SUCCESS\_CONNECTIONS .- FALSE.
8. FW\_PROFILE\_CONFIG\_LOG\_IGNORED\_RULES .- TRUE.
9. FW\_PROFILE\_CONFIG\_LOG\_MAX\_FILE\_SIZE .- 1024.
10. FW\_PROFILE\_CONFIG\_LOG\_FILE\_PATH .- L"".
11. FW\_PROFILE\_CONFIG\_DISABLE\_INBOUND\_NOTIFICATIONS .- FALSE.
12. FW\_PROFILE\_CONFIG\_AUTH\_APPS\_ALLOW\_USER\_PREF\_MERGE .- TRUE.
13. FW\_PROFILE\_CONFIG\_GLOBAL\_PORTS\_ALLOW\_USER\_PREF\_MERGE .- TRUE.
14. FW\_PROFILE\_CONFIG\_ALLOW\_LOCAL\_POLICY\_MERGE .- TRUE.
15. FW\_PROFILE\_CONFIG\_ALLOW\_LOCAL\_IPSEC\_POLICY\_MERGE .- TRUE.
16. FW\_PROFILE\_CONFIG\_DISABLED\_INTERFACES .- {0}.
17. FW\_PROFILE\_CONFIG\_DEFAULT\_OUTBOUND\_ACTION .- 0 (0 is allow).
18. FW\_PROFILE\_CONFIG\_DEFAULT\_INBOUND\_ACTION.- 1 (1 is block).

Global configuration options default values:

1. FW\_GLOBAL\_CONFIG\_POLICY\_VERSION\_SUPPORTED .- 0x0200
2. on Windows Vista.
3. FW\_GLOBAL\_CONFIG\_POLICY\_VERSION\_SUPPORTED .- 0x0201
4. on Windows Vista SP1 and Windows Server 2008.
5. FW\_GLOBAL\_CONFIG\_CURRENT\_PROFILE .- FW\_PROFILE\_TYPE\_PUBLIC.
6. FW\_GLOBAL\_CONFIG\_DISABLE\_STATEFUL\_FTP .- FALSE.
7. FW\_GLOBAL\_CONFIG\_DISABLE\_STATEFUL\_PPTP .- FALSE.
8. FW\_GLOBAL\_CONFIG\_SA\_IDLE\_TIME .- 300.
9. FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING
10. .- FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING\_UTF\_8.
11. FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT
12. .- FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_NEIGHBOR\_DISC.
13. FW\_GLOBAL\_CONFIG\_CRL\_CHECK .- 0.
14. FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT
15. .- FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_SERVER\_BEHIND\_NAT.
16. FW\_GLOBAL\_CONFIG\_POLICY\_VERSION .- 0x0200.
17. FW\_GLOBAL\_CONFIG\_BINARY\_VERSION\_SUPPORTED .- 0x201. This value is
18. present only in Windows Vista SP1 and Windows Server 2008.

[<11> Section 2.2.54](#Appendix_A_Target_11): Windows uses the three fields of the FW\_OS\_PLATFORM data type to identify Windows platform types. The fields in this data type correspond to the fields of the Windows OSVERSIONINFOEX data type (for more information, see [MSDN-OSVERSIONINFOEX]). The **bPlatform** field in this specification corresponds to the **dwPlatformId** field in MSDN. The **bMajorVersion** field in this specification corresponds to the **dwMajorVersion** field in MSDN. The **bMinorVersion** field in this specification corresponds to the **dwMinorVersion** field in MSDN. The Windows firewall and advanced security components extract the OSVERSIONINFOEX values and use them to enforce PlatformValidityList conditions in FW\_RULE (section 2.2.36) and FW\_CS\_RULE (section 2.2.54) rules.

[<12> Section 2.2.54](#Appendix_A_Target_12): On Windows 7 and Windows Server 2008 R2 the **wszRuleId** size is less than 512 characters. On Windows Vista and Windows Server 2008 it is less than 1000 characters.

[<13> Section 2.2.54](#Appendix_A_Target_13): On Windows 7 and Windows Server 2008 R2 the **wszPhase1AuthSet**, **wszPhase2AuthSet**, and **wszPhase2CryptoSet** sizes are less than 255 characters. On Windows Vista and Windows Server 2008 they are less than 1000 characters.

[<14> Section 2.2.63](#Appendix_A_Target_14): On Windows Vista and Windows Server 2008, the only duplicate check performed is for the anonymous method.

[<15> Section 2.2.63](#Appendix_A_Target_15): On Windows Vista and Windows Server 2008, the only duplicate check performed is for the anonymous method.

[<16> Section 2.2.64](#Appendix_A_Target_16): On Windows Vista and Windows Server 2008, the only duplicate check performed is for the anonymous method.

[<17> Section 2.2.64](#Appendix_A_Target_17): On Windows Vista and Windows Server 2008, the only duplicate check performed is for the anonymous method.

[<18> Section 2.2.82](#Appendix_A_Target_18): Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2 set **TransportFilterId** to the filter key of the Windows Filtering Platform filter used to enforce the [**security association**](#gt_67cbf867-7a49-41f3-a68f-37b5f9035acb) (for more information, see [[MSWFPSDK]](https://go.microsoft.com/fwlink/?LinkId=90220)).

[<19> Section 2.2.84](#Appendix_A_Target_19): Windows uses the three fields of the FW\_OS\_PLATFORM data type to identify Windows platform types. The fields in this data type correspond to the fields of the Windows OSVERSIONINFOEX data type (for more information, see [MSDN-OSVERSIONINFOEX]). The **bPlatform** field in this specification corresponds to the **dwPlatformId** field in MSDN. The **bMajorVersion** field in this specification corresponds to the **dwMajorVersion** field in MSDN. The **bMinorVersion** field in this specification corresponds to the **dwMinorVersion** field in MSDN. The Windows firewall and advanced security components extract the OSVERSIONINFOEX values and use them to enforce PlatformValidityList conditions in FW\_RULE (section 2.2.36) and FW\_CS\_RULE (section 2.2.54) rules.

[<20> Section 2.2.95](#Appendix_A_Target_20): By default, Windows uses the IKEv1 and [**AuthIP**](#gt_3791f3e1-cf2f-4605-9fcc-54f526f036cf) keying modules.

[<21> Section 2.2.96](#Appendix_A_Target_21): In schema version 0x0214, the value for the FW\_TRUST\_TUPLE\_KEYWORD\_MAX flag is 0x0004.

[<22> Section 2.2.102](#Appendix_A_Target_22): In Windows, audit events that are generated by rules that specify the FW\_RULE\_FLAGS2\_CALLOUT\_AND\_AUDIT flag are sent to the audit event log.

[<23> Section 3.1.3](#Appendix_A_Target_23): During server initialization, Windows uses default values to initialize the Phase 1 and Phase 2 primary **AuthenticationSet** objects if these objects are not already present in **LocalStore** or **GroupPolicyRSoPStore**. The same defaults are used for both **LocalStore** and **GroupPolicyRSoPStore**. These defaults are as follows:

1. #define FW\_DEFAULT\_P1\_PRIMARY\_AUTH\_SET\_NAME\_STR
2. L"Default Phase1 Primary AuthSet"
3. #define FW\_DEFAULT\_P2\_PRIMARY\_AUTH\_SET\_NAME\_STR
4. L"Default Phase2 Primary AuthSet"
5. #define RTL\_NUMBER\_OF(A) (sizeof(A)/sizeof((A)[0]))
6. FW\_AUTH\_SUITE g\_DefaultPrimaryAuthSuitePhase1[] =
7. {
8. { FW\_AUTH\_METHOD\_MACHINE\_KERB, {0} }
9. };
10. FW\_AUTH\_SET g\_DefaultPrimaryAuthSetPhase1 =
11. {
12. NULL,
13. 0x0200,
14. FW\_IPSEC\_PHASE\_1,
15. L"{E5A5D32A-4BCE-4e4d-B07F-4AB1BA7E5FE3}",
16. FW\_DEFAULT\_P1\_PRIMARY\_AUTH\_SET\_NAME\_STR,
17. FW\_DEFAULT\_P1\_PRIMARY\_AUTH\_SET\_NAME\_STR,
18. NULL,
19. RTL\_NUMBER\_OF(g\_DefaultPrimaryAuthSuitePhase1),
20. g\_DefaultPrimaryAuthSuitePhase1,
21. FW\_RULE\_ORIGIN\_HARDCODED,
22. NULL,
23. FW\_RULE\_STATUS\_OK,
24. 0
25. };
26. FW\_AUTH\_SET g\_DefaultPrimaryAuthSetPhase2 =
27. {
28. NULL,
29. 0x0200,
30. FW\_IPSEC\_PHASE\_2,
31. L"{E5A5D32A-4BCE-4e4d-B07F-4AB1BA7E5FE4}",
32. FW\_DEFAULT\_P2\_PRIMARY\_AUTH\_SET\_NAME\_STR,
33. FW\_DEFAULT\_P2\_PRIMARY\_AUTH\_SET\_NAME\_STR,
34. NULL,
35. 0,
36. NULL,
37. FW\_RULE\_ORIGIN\_HARDCODED,
38. NULL,
39. FW\_RULE\_STATUS\_OK,
40. 0
41. };

[<24> Section 3.1.3](#Appendix_A_Target_24): During server initialization, Windows uses default values to initialize the Phase 1 and Phase 2 primary **CryptoSet** objects if these objects are not already present in **LocalStore** or **GroupPolicyRSoPStore**. The same defaults are used for both **LocalStore** and **GroupPolicyRSoPStore**. These defaults are as follows:

1. #define FW\_DEFAULT\_P1\_PRIMARY\_CRYPTO\_SET\_NAME\_STR
2. L"Default Phase1 Primary CryptoSet"
3. #define FW\_DEFAULT\_P2\_PRIMARY\_CRYPTO\_SET\_NAME\_STR
4. L"Default Phase2 Primary CryptoSet"
5. FW\_PHASE1\_CRYPTO\_SUITE g\_DefaultPrimaryCryptoSuitesPhase1[] =
6. {
7. {FW\_CRYPTO\_KEY\_EXCHANGE\_DH2,
8. FW\_CRYPTO\_ENCRYPTION\_AES128,
9. FW\_CRYPTO\_HASH\_SHA1},
10. {FW\_CRYPTO\_KEY\_EXCHANGE\_DH2,
11. FW\_CRYPTO\_ENCRYPTION\_3DES,
12. FW\_CRYPTO\_HASH\_SHA1}
13. };
15. FW\_CRYPTO\_SET g\_DefaultPrimaryCryptoSetPhase1 =
16. {
17. NULL,
18. 0x0200,
19. FW\_IPSEC\_PHASE\_1,
20. L"{E5A5D32A-4BCE-4e4d-B07F-4AB1BA7E5FE1}",
21. FW\_DEFAULT\_P1\_PRIMARY\_CRYPTO\_SET\_NAME\_STR,
22. FW\_DEFAULT\_P1\_PRIMARY\_CRYPTO\_SET\_NAME\_STR,
23. NULL,
24. {
25. 0, //flags
26. 0, //RTL\_NUMBER\_OF(g\_DefaultPrimaryCryptoSuitesPhase1),
27. 0, //g\_DefaultPrimaryCryptoSuitesPhase1,
28. 0, // 480,
29. 0
30. },
31. FW\_RULE\_ORIGIN\_HARDCODED,
32. NULL,
33. FW\_RULE\_STATUS\_OK,
34. 0
35. };
37. FW\_PHASE2\_CRYPTO\_SUITE g\_DefaultPrimaryCryptoSuitesPhase2[] =
38. {
39. {FW\_CRYPTO\_PROTOCOL\_ESP,
40. FW\_CRYPTO\_HASH\_NONE,
41. FW\_CRYPTO\_HASH\_SHA1,
42. FW\_CRYPTO\_ENCRYPTION\_NONE,
43. FW\_DEFAULT\_CRYPTO\_PHASE2\_TIMEOUT\_MINUTES,
44. FW\_DEFAULT\_CRYPTO\_PHASE2\_TIMEOUT\_KBYTES},
45. {FW\_CRYPTO\_PROTOCOL\_ESP,
46. FW\_CRYPTO\_HASH\_NONE,
47. FW\_CRYPTO\_HASH\_SHA1,
48. FW\_CRYPTO\_ENCRYPTION\_AES128,
49. FW\_DEFAULT\_CRYPTO\_PHASE2\_TIMEOUT\_MINUTES,
50. FW\_DEFAULT\_CRYPTO\_PHASE2\_TIMEOUT\_KBYTES},
51. {FW\_CRYPTO\_PROTOCOL\_ESP,
52. FW\_CRYPTO\_HASH\_NONE,
53. FW\_CRYPTO\_HASH\_SHA1,
54. FW\_CRYPTO\_ENCRYPTION\_3DES,
55. FW\_DEFAULT\_CRYPTO\_PHASE2\_TIMEOUT\_MINUTES,
56. FW\_DEFAULT\_CRYPTO\_PHASE2\_TIMEOUT\_KBYTES},
57. {FW\_CRYPTO\_PROTOCOL\_AH,
58. FW\_CRYPTO\_HASH\_SHA1,
59. FW\_CRYPTO\_HASH\_NONE,
60. FW\_CRYPTO\_ENCRYPTION\_NONE,
61. FW\_DEFAULT\_CRYPTO\_PHASE2\_TIMEOUT\_MINUTES,
62. FW\_DEFAULT\_CRYPTO\_PHASE2\_TIMEOUT\_KBYTES}
63. };
65. FW\_CRYPTO\_SET g\_DefaultPrimaryCryptoSetPhase2 =
66. {
67. NULL,
68. 0x0200,
69. FW\_IPSEC\_PHASE\_2,
70. L"{E5A5D32A-4BCE-4e4d-B07F-4AB1BA7E5FE2}",
71. FW\_DEFAULT\_P2\_PRIMARY\_CRYPTO\_SET\_NAME\_STR,
72. FW\_DEFAULT\_P2\_PRIMARY\_CRYPTO\_SET\_NAME\_STR,
73. NULL,
74. {
75. {
76. 0, // FW\_PHASE2\_CRYPTO\_PFS\_DISABLE,
77. 0, // RTL\_NUMBER\_OF(g\_DefaultPrimaryCryptoSuitesPhase2),
78. 0, // g\_DefaultPrimaryCryptoSuitesPhase2
79. }
80. },
81. FW\_RULE\_ORIGIN\_HARDCODED,
82. NULL,
83. FW\_RULE\_STATUS\_OK,
84. 0
85. };
86. void FwDefaultPrimaryCryptoSetsInit()
87. {
88. // Init Phase 1 Crypto.
89. g\_DefaultPrimaryCryptoSetPhase1.dwNumPhase1Suites =
90. RTL\_NUMBER\_OF(g\_DefaultPrimaryCryptoSuitesPhase1);
91. g\_DefaultPrimaryCryptoSetPhase1.pPhase1Suites =
92. g\_DefaultPrimaryCryptoSuitesPhase1;
93. g\_DefaultPrimaryCryptoSetPhase1.dwTimeOutMinutes = 480;
94. //Init Phase 2 Crypto
95. g\_DefaultPrimaryCryptoSetPhase2.Pfs =
96. FW\_PHASE2\_CRYPTO\_PFS\_DISABLE;
97. g\_DefaultPrimaryCryptoSetPhase2.dwNumPhase2Suites =
98. RTL\_NUMBER\_OF(g\_DefaultPrimaryCryptoSuitesPhase2);
99. g\_DefaultPrimaryCryptoSetPhase2.pPhase2Suites =
100. g\_DefaultPrimaryCryptoSuitesPhase2;
101. }

[<25> Section 3.1.3](#Appendix_A_Target_25): Windows selects a default value for the **ProfileConfiguration** option and the **GlobalConfiguration** option. These configuration default values are secure, and it is recommended to use these values as default values. **ProfileConfiguration** option default values:

1. FW\_PROFILE\_CONFIG\_ENABLE\_FW .- TRUE.
2. FW\_PROFILE\_CONFIG\_DISABLE\_STEALTH\_MODE .- FALSE.
3. FW\_PROFILE\_CONFIG\_SHIELDED .- FALSE.
4. FW\_PROFILE\_CONFIG\_DISABLE\_UNICAST\_RESPONSES\_TO\_MULTICAST\_BROADCAST
5. .- FALSE.
6. FW\_PROFILE\_CONFIG\_LOG\_DROPPED\_PACKETS .- FALSE.
7. FW\_PROFILE\_CONFIG\_LOG\_SUCCESS\_CONNECTIONS .- FALSE.
8. FW\_PROFILE\_CONFIG\_LOG\_IGNORED\_RULES .- TRUE.
9. FW\_PROFILE\_CONFIG\_LOG\_MAX\_FILE\_SIZE .- 1024.
10. FW\_PROFILE\_CONFIG\_LOG\_FILE\_PATH .- L"".
11. FW\_PROFILE\_CONFIG\_DISABLE\_INBOUND\_NOTIFICATIONS .- FALSE.
12. FW\_PROFILE\_CONFIG\_AUTH\_APPS\_ALLOW\_USER\_PREF\_MERGE .- TRUE.
13. FW\_PROFILE\_CONFIG\_GLOBAL\_PORTS\_ALLOW\_USER\_PREF\_MERGE .- TRUE.
14. FW\_PROFILE\_CONFIG\_ALLOW\_LOCAL\_POLICY\_MERGE .- TRUE.
15. FW\_PROFILE\_CONFIG\_ALLOW\_LOCAL\_IPSEC\_POLICY\_MERGE .- TRUE.
16. FW\_PROFILE\_CONFIG\_DISABLED\_INTERFACES .- {0}.
17. FW\_PROFILE\_CONFIG\_DEFAULT\_OUTBOUND\_ACTION .- 0 (0 is allow).
18. FW\_PROFILE\_CONFIG\_DEFAULT\_INBOUND\_ACTION.- 1 (1 is block).

**GlobalConfiguration** options default values:

1. FW\_GLOBAL\_CONFIG\_POLICY\_VERSION\_SUPPORTED .- 0x0200
2. on Windows Vista.
3. FW\_GLOBAL\_CONFIG\_POLICY\_VERSION\_SUPPORTED .- 0x0201
4. on Windows Vista SP1 and Windows Server 2008.
5. FW\_GLOBAL\_CONFIG\_CURRENT\_PROFILE .- FW\_PROFILE\_TYPE\_PUBLIC.
6. FW\_GLOBAL\_CONFIG\_DISABLE\_STATEFUL\_FTP .- FALSE.
7. FW\_GLOBAL\_CONFIG\_DISABLE\_STATEFUL\_PPTP .- FALSE.
8. FW\_GLOBAL\_CONFIG\_SA\_IDLE\_TIME .- 300.
9. FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING
10. .- FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING\_UTF\_8.
11. FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT
12. .- FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_NEIGHBOR\_DISC.
13. FW\_GLOBAL\_CONFIG\_CRL\_CHECK .- 0.
14. FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT
15. .- FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_SERVER\_BEHIND\_NAT.
16. FW\_GLOBAL\_CONFIG\_POLICY\_VERSION .- 0x0200.
17. FW\_GLOBAL\_CONFIG\_BINARY\_VERSION\_SUPPORTED .- 0x201. This value is
18. present only in Windows Vista SP1 and Windows Server 2008.

[<26> Section 3.1.4](#Appendix_A_Target_26): In Windows Vista, Windows Server 2008, Windows 7, and Windows Server 2008 R2, security principals are identified by [**SIDs**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) (see [[MS-DTYP]](%5bMS-DTYP%5d.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.2). The authorized clients are represented by the S-1-5-32-544 and the S-1-5-32-556 SIDs. If the client's identity token (see [MS-DTYP] section 2.5.2) does not contain at least one of these SIDs, the server fails the call.

[<27> Section 3.1.4.6](#Appendix_A_Target_27): Path validations were not performed in Windows Vista and Windows Server 2008 at edit time.

[<28> Section 3.1.4.47](#Appendix_A_Target_28): Path validations were not performed in Windows Vista and Windows Server 2008 at edit time.

[<29> Section 3.1.6.5](#Appendix_A_Target_29): Windows determines whether it is operating in [**common criteria mode**](#gt_52549a11-2432-4a5c-966f-5f8a32de9162) by calling the BCryptGetFipsAlgorithmMode API. For more information, see [[MSDN-BCryptGetFipsAlgorithmMode]](https://go.microsoft.com/fwlink/?LinkId=211797).

[<30> Section 3.1.6.6](#Appendix_A_Target_30): Windows enforces the effective firewall policy by converting the settings to Windows Filtering Platform filters. For more information, see [MSWFPSDK].

# 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_6f1e3ed281e44ec0ad6b1012125aa636) 219

[server](#section_43507d538955416db913dfb27dc76b17) 117

[Adding a firewall rule example](#section_511137736df94bcdbe96d262ad5f1cbd) 221

[Adding firewall rule example](#section_511137736df94bcdbe96d262ad5f1cbd) 221

[Applicability](#section_2c679c7944d546fe8f3d5b5efaf2a7b3) 16

C

[Capability negotiation](#section_6b14b8f991ab4f09873f9e2334196cdc) 16

[Change tracking](#section_acbbce890ed343c386f0f26b9285863c) 296

Client

[abstract data model](#section_6f1e3ed281e44ec0ad6b1012125aa636) 219

[initialization](#section_27cba968dfb34c91b36b1e615550eaa1) 220

[local events](#section_425b61644e56476b84a1678721baa789) 220

[message processing](#section_0bf6ef331e1548d081172f79d2f21e5c) 220

[sequencing rules](#section_0bf6ef331e1548d081172f79d2f21e5c) 220

[timer events](#section_ffc97905f3ca4bf5aa1cbed456765054) 220

[timers](#section_38e9cda4e67b4be483d61c3dc7ca7b4b) 220

[Closing a policy store handle example](#section_2959ec3adf7c4da9b118385b62312067) 223

[Closing policy store example](#section_2959ec3adf7c4da9b118385b62312067) 223

[Common data types](#section_2f45305aaf9249678d918f137ebc11e0) 18

D

Data model - abstract

[client](#section_6f1e3ed281e44ec0ad6b1012125aa636) 219

[server](#section_43507d538955416db913dfb27dc76b17) 117

Data types

[common - overview](#section_2f45305aaf9249678d918f137ebc11e0) 18

[Data types - common](#section_2f45305aaf9249678d918f137ebc11e0) 18

E

[Enumerating firewall rules example](#section_36cddff4c4274863a58d3d913a12b221) 223

[Enumerating the firewall rules example](#section_36cddff4c4274863a58d3d913a12b221) 223

Events

[local - client](#section_425b61644e56476b84a1678721baa789) 220

[local - server](#section_6e5c3ec0c748485687c99d6b35ab7280) 217

[timer - client](#section_ffc97905f3ca4bf5aa1cbed456765054) 220

[timer - server](#section_d55935b3e16c4bbb8ea6a89559a14509) 217

Examples

[adding a firewall rule](#section_511137736df94bcdbe96d262ad5f1cbd) 221

[adding firewall rule example](#section_511137736df94bcdbe96d262ad5f1cbd) 221

[closing a policy store handle](#section_2959ec3adf7c4da9b118385b62312067) 223

[closing policy store example](#section_2959ec3adf7c4da9b118385b62312067) 223

[enumerating firewall rules example](#section_36cddff4c4274863a58d3d913a12b221) 223

[enumerating the firewall rules](#section_36cddff4c4274863a58d3d913a12b221) 223

[opening a policy store](#section_230d1ae7b42e4d9cb997b1463aaa0ded) 221

[opening policy store example](#section_230d1ae7b42e4d9cb997b1463aaa0ded) 221

F

[Fields - vendor-extensible](#section_d8ff8b08554f4c13805f785583a16e78) 17

[Full IDL](#section_1503b9d77fec479399726ad58720c9db) 226

[FW\_ADAPTER structure](#section_24178f7ad5f749bda7ee69fe2b02ff4f) 64

[FW\_ADDRESS\_KEYWORD enumeration](#section_d69ec3fe85074524bdcc813cbb3bf85f) 27

[FW\_ADDRESSES structure](#section_67b07a7f8748487e8fd8c75fe3b307e6) 28

[FW\_AUTH\_INFO structure](#section_cd08ebed295d4c7786c64ba87aa4df00) 97

[FW\_AUTH\_METHOD enumeration](#section_59e71d46440747d29cf58889fd3a74f2) 76

[FW\_AUTH\_SET structure](#section_bbec7ab70a1d49a49c8ea4784bbe6f16) 83

[FW\_AUTH\_SET\_FLAGS enumeration](#section_63a4848affba477a826dcd616627a0fa) 111

[FW\_AUTH\_SET2\_10 structure](#section_c1acee0a747a4482b1590c96ca57fe29) 80

[FW\_AUTH\_SUITE structure](#section_447c3a4c9543495dab7db7ca639712ef) 79

[FW\_AUTH\_SUITE\_FLAGS enumeration](#section_ff03b48b87cf4020bf0b7f4c680d9b02) 77

[FW\_AUTH\_SUITE2\_10 structure](#section_bae6ee78bedc4a3a95458920c2a93cda) 78

[FW\_BYTE\_BLOB structure](#section_8e7e10f0590043618403d8833934c491) 95

[FW\_CERT\_CRITERIA structure](#section_8f4e2f51826f43c794055395b824bd41) 75

[FW\_CERT\_CRITERIA\_FLAGS enumeration](#section_df8aba33578c4f01802c435306e63409) 75

[FW\_CERT\_CRITERIA\_NAME\_TYPE enumeration](#section_b83533eb9e4142558dd4be8bf75d1053) 74

[FW\_CERT\_CRITERIA\_TYPE enumeration](#section_84a1305fff3a4bbeba81e1c111ff5ba4) 74

[FW\_CERT\_INFO structure](#section_dfc16343eb514bbab890df5a58839b2b) 96

[FW\_CONFIG\_FLAGS enumeration](#section_eb0dce3b08d34c32b9fec8bdaf8e256a) 64

[FW\_COOKIE\_PAIR structure](#section_1974c8d425694bcea0cbbc692963fca1) 96

[FW\_CRYPTO\_ENCRYPTION\_TYPE enumeration](#section_a74eafcd19ff4305b1050952ae20a2dd) 87

[FW\_CRYPTO\_HASH\_TYPE enumeration](#section_eae09dba8bed43e385bddce8497de9a1) 88

[FW\_CRYPTO\_KEY\_EXCHANGE\_TYPE enumeration](#section_b400008796b046c7b15209a55d985f0f) 86

[FW\_CRYPTO\_PROTOCOL\_TYPE enumeration](#section_d97bac3603e842159984f2fbddd66be0) 89

[FW\_CRYPTO\_SET structure](#section_a468fe9e113b4155a63d0db3aac12619) 92

[FW\_CRYPTO\_SET\_FLAGS enumeration](#section_c5990757c8ec44f39fd6f8429b3ba0a4) 111

[FW\_CS\_RULE structure](#section_0d0641105f2e4b68aa63032c6cd5e4c6) 70

[FW\_CS\_RULE\_ACTION enumeration](#section_358add6dadf74b6f8b6a66535ac802ed) 68

[FW\_CS\_RULE\_FLAGS enumeration](#section_567fdc537b1c418b8b11dd7267f75bad) 67

[FW\_CS\_RULE2\_0 structure](#section_18277e4c628d44239fe3ba98601ba50d) 69

[FW\_CS\_RULE2\_10 structure](#section_8f1281f0bb53491fa552742a1a922137) 68

[FW\_DATA\_TYPE enumeration](#section_cddfd96ff9f34ae2ac497ebc2e6ae44e) 104

[FW\_DIAG\_APP structure](#section_9d423f51a0c044998df6cba099fc4b27) 65

[FW\_DIRECTION enumeration](#section_34806362c31f4531ad8c5a43d5870223) 26

[FW\_ENDPOINTS structure](#section_2318781e64bc475d997502cc9d56ef0b) 97

[FW\_ENFORCEMENT\_STATE enumeration](#section_4767a26e06874c4fad948de700e5cf83) 43

[FW\_ENUM\_RULES\_FLAGS enumeration](#section_aff3633c43b24cefba63f35b204a8a08) 48

[FW\_GLOBAL\_CONFIG enumeration](#section_faf4ffbe1d5140adae902230f2c0b6a9) 61

[FW\_GLOBAL\_CONFIG\_IPSEC\_EXEMPT\_VALUES enumeration](#section_7daabd9f74c34295add6e2402b01b191) 59

[FW\_GLOBAL\_CONFIG\_IPSEC\_THROUGH\_NAT\_VALUES enumeration](#section_69a0da19647041958a32a204566fa813) 60

[FW\_GLOBAL\_CONFIG\_PRESHARED\_KEY\_ENCODING\_VALUES enumeration](#section_b9d24a5e77554c60adebe0c7a718f909) 60

[FW\_ICMP\_TYPE\_CODE structure](#section_6feb51f6495e4396b86c0dbfc5650dff) 25

[FW\_ICMP\_TYPE\_CODE\_LIST structure](#section_efce15b5aabb4273abce078aca9c6427) 25

[FW\_INTERFACE\_LUIDS structure](#section_ea420d0f03ed48e5b786621db56419d5) 26

[FW\_INTERFACE\_TYPE enumeration](#section_d067c0284ea943199ddf0fb51c191b35) 26

[FW\_IP\_VERSION enumeration](#section_f0350f85904844268264fe11d62c5af3) 66

[FW\_IPSEC\_PHASE enumeration](#section_e23d75abc5f14b77b290f0e0e25d72e1) 66

[FW\_IPV4\_ADDRESS\_RANGE structure](#section_f11b849900a043d8bf3a6b596ec0d55c) 22

[FW\_IPV4\_RANGE\_LIST structure](#section_de836948bf68424d906d406723bd0deb) 22

[FW\_IPV4\_SUBNET structure](#section_ed0a65ecd3b6461ca4caecf08ed18025) 21

[FW\_IPV4\_SUBNET\_LIST structure](#section_a230f0e48005498389e3e417b3354ae5) 21

[FW\_IPV6\_ADDRESS\_RANGE structure](#section_fbf349cc8455490db97d951c63084861) 22

[FW\_IPV6\_RANGE\_LIST structure](#section_062373f2c3dc49a98b3f960d2a88332d) 23

[FW\_IPV6\_SUBNET structure](#section_ccb501eb66204b49befd2b59828a28c8) 21

[FW\_IPV6\_SUBNET\_LIST structure](#section_a8d05086bc48453eaebfaa11fb95b9bd) 21

[FW\_KEY\_MODULE enumeration](#section_f309dd04f6a4436eaf899fbc7a3aa3ce) 108

[FW\_MATCH\_KEY enumeration](#section_d8d31780181b44c5ac1a51e516ba97ce) 103

[FW\_MATCH\_TYPE enumeration](#section_47c00878fbea43d0a79300ebf94ae400) 105

[FW\_MATCH\_VALUE structure](#section_3af7dd01b8c648b0b179824a10464911) 105

[FW\_MM\_RULE structure](#section_4b3fc163fede434c90cb557992caa5da) 101

[FW\_NETWORK structure](#section_682c407a18024e9eb96a8c74315e5746) 64

[FW\_NETWORK\_NAMES structure](#section_1e72af2f869f484ab1478371ce830a0c) 111

[FW\_OBJECT\_CTRL\_FLAG enumeration](#section_2b31e26d78864993b2483df3bee941fa) 43

[FW\_OBJECT\_METADATA structure](#section_cecfb58e2c794b7db92c7666fa06d559) 45

[FW\_OS\_PLATFORM structure](#section_aac2be8cf4ab4b0a995774462137fd8a) 46

[FW\_OS\_PLATFORM\_LIST structure](#section_dae582a2831e4e6a8dc923be7358fefd) 47

[FW\_OS\_PLATFORM\_OP enumeration](#section_925113284eed4764a680d2d35214d051) 45

[FW\_PHASE1\_CRYPTO\_FLAGS enumeration](#section_6f3d76aab2424f2b8c5ff9ccbbc04c53) 91

[FW\_PHASE1\_CRYPTO\_SUITE structure](#section_cd273bb434214831a979347630eb3b16) 89

[FW\_PHASE1\_KEY\_MODULE\_TYPE enumeration](#section_f0944fb9dc8741128b5fe897027f2b73) 96

[FW\_PHASE1\_SA\_DETAILS structure](#section_218b010dea7f4b14b728aa27dc418236) 98

[FW\_PHASE2\_CRYPTO\_PFS enumeration](#section_a281cd9c0d3e40718a4fd0b5ada838a2) 91

[FW\_PHASE2\_CRYPTO\_SUITE structure](#section_99a997b6de524648977ce9efd6058254) 90

[FW\_PHASE2\_SA\_DETAILS structure](#section_e6091d0ae57846e08656113d1e1e360d) 99

[FW\_PHASE2\_TRAFFIC\_TYPE enumeration](#section_2b2a975cd6124943a3314b7076944d61) 99

[FW\_POLICY\_ACCESS\_RIGHT enumeration](#section_fd654514982846a29935e987ef64c860) 20

[FW\_PORT\_KEYWORD enumeration](#section_f2a500d8b8cb48e1b4f21f85c89011ad) 24

[FW\_PORT\_RANGE structure](#section_3122d4b1950043af900a27fe64688623) 23

[FW\_PORT\_RANGE\_LIST structure](#section_5d0e412713fe46b38685f7784941c5b4) 23

[FW\_PORTS structure](#section_df16981d5d474a25aed73d03cf33998a) 25

[FW\_PRODUCT structure](#section_969d38731e424dc48559f5c0c9caa965) 65

[FW\_PROFILE\_CONFIG enumeration](#section_5a6e0d39802d456bb483c7360566fcdd) 57

[FW\_PROFILE\_TYPE enumeration](#section_7704e238174d4a5eb8095f3787dd8acc) 19

[FW\_QUERY structure](#section_51b69145317a4368a6e57159e946ff14) 107

[FW\_QUERY\_CONDITION structure](#section_fa19141ffc3c45159555b6db62c475aa) 106

[FW\_QUERY\_CONDITIONS structure](#section_98765a8c9a0c471bb2ee56c931ea5604) 107

[FW\_RULE structure](#section_8c008258166d46d49090f2ffaa01be4b) 52

[FW\_RULE\_ACTION enumeration](#section_702e3c23c9d843db8380b4c670dd7f7d) 49

[FW\_RULE\_CATEGORY enumeration](#section_16b2c1a3789f4cdeb3d1ecd9182b60fe) 65

[FW\_RULE\_FLAGS enumeration](#section_fc67ea0419f04ccd8912abe467e7c11a) 49

[FW\_RULE\_ORIGIN\_TYPE enumeration](#section_9d295321d75c41c0ab0d7a78df40f77c) 47

[FW\_RULE\_STATUS enumeration](#section_c4fd81e638dd47c58a0fa98a11cf190e) 29

[FW\_RULE\_STATUS\_CLASS enumeration](#section_81877aa634c74664adf4ff8d5e0c0bc6) 42

[FW\_RULE2\_0 structure](#section_af151922b0924e5c9b694dca6b6d2ffc) 51

[FW\_STORE\_TYPE enumeration](#section_37ebed958abf472c8b4b7a510a2a6baa) 18

[FW\_TRUST\_TUPLE\_KEYWORD enumeration](#section_f6af7e074b9949fb81b9f9e2abf5a508) 109

G

[Glossary](#section_d891304d92d84f9f9e60a1f02b28ae30) 9

I

[IDL](#section_1503b9d77fec479399726ad58720c9db) 226

[Implementer - security considerations](#section_41e32b09744241898b7717ded87b4751) 225

[Index of security parameters](#section_ea9a532ff18a4d32a65413182cb99338) 225

[Informative references](#section_cbdceb4d03bb4b918c70f2ec74fad0bc) 12

Initialization

[client](#section_27cba968dfb34c91b36b1e615550eaa1) 220

[server](#section_e8924ac5aa4a41d1bf654f46b3d399aa) 120

[Introduction](#section_495b9c3d12654065bc81b0c83bf14bc9) 9

L

Local events

[client](#section_425b61644e56476b84a1678721baa789) 220

[server](#section_6e5c3ec0c748485687c99d6b35ab7280) 217

M

Message processing

[client](#section_0bf6ef331e1548d081172f79d2f21e5c) 220

[server](#section_b0c9335283714d63abcab0cc8dbcc3d7) 121

Messages

[common data types](#section_2f45305aaf9249678d918f137ebc11e0) 18

[transport](#section_81eb95d6df6349b6905265c99664e71f) 18

Methods

[RRPC\_FWAddAuthenticationSet (Opnum 17)](#section_c1dbe17a32cf4910843b7d2690e9087a) 146

[RRPC\_FWAddAuthenticationSet2\_10 (Opnum 52)](#section_25bb2543bda546e5ba7afa6c7a356ba4) 182

[RRPC\_FWAddAuthenticationSet2\_20 (Opnum 62)](#section_b098dcdb7058470dba9df042dec221c0) 192

[RRPC\_FWAddConnectionSecurityRule (Opnum 12)](#section_4e31fcb5b70148b58a45d70bf85281ab) 141

[RRPC\_FWAddConnectionSecurityRule2\_10 (Opnum 49)](#section_b7a0d5cca02f46078681e8b5589d3451) 179

[RRPC\_FWAddConnectionSecurityRule2\_20 (Opnum 58)](#section_222c230acb4c41b2ac44a770cd71d66c) 188

[RRPC\_FWAddCryptoSet (Opnum 22)](#section_0399561877894a00828cd8c497b643e3) 151

[RRPC\_FWAddCryptoSet2\_10 (Opnum 55)](#section_604883c63e674140bec0ea68adf4eb53) 185

[RRPC\_FWAddFirewallRule (Opnum 5)](#section_4d7baada78fd469a9c292885e5d3b5d0) 134

[RRPC\_FWAddFirewallRule2\_10 (Opnum 46)](#section_5ab52f049e9d426b854560e2367fdffd) 175

[RRPC\_FWAddFirewallRule2\_20 (Opnum 66)](#section_d637fee2b28d4cea8005303347de94f7) 196

[RRPC\_FWAddFirewallRule2\_24 (Opnum 70)](#section_778f196ee6f347bfa3a6f6f5f7660b21) 200

[RRPC\_FWAddFirewallRule2\_25 (Opnum 74)](#section_fb298ceac5fe4609b71ce0221dde97c1) 205

[RRPC\_FWAddFirewallRule2\_26 (Opnum 78)](#section_944db4f78e434b1b9060d9bd4ef8e7d1) 209

[RRPC\_FWAddFirewallRule2\_27 (Opnum 82)](#section_980933b01319416a94ee374703462728) 213

[RRPC\_FWAddMainModeRule (Opnum 32)](#section_40973e9f20c84e7b8f96f2664a622f03) 162

[RRPC\_FWClosePolicyStore (Opnum 1)](#section_203dc96f2e74449aae18aaae0689c092) 130

[RRPC\_FWDeleteAllAuthenticationSets (Opnum 20)](#section_7d8f82f7d9284ebc8afc576751e2e8d2) 149

[RRPC\_FWDeleteAllConnectionSecurityRules (Opnum 15)](#section_ee4f85d9c07449658508c8ec96ae3b77) 144

[RRPC\_FWDeleteAllCryptoSets (Opnum 25)](#section_67cf747ee0054630932928887df1e09d) 155

[RRPC\_FWDeleteAllFirewallRules (Opnum 8)](#section_ca2da2f2b8ef4981abbafed5c713990d) 136

[RRPC\_FWDeleteAllMainModeRules (Opnum 35)](#section_d47dfd57c6b9448eb5a0127d3f87cd38) 164

[RRPC\_FWDeleteAuthenticationSet (Opnum 19)](#section_b143f0f85f274595b5310197fc491d0b) 148

[RRPC\_FWDeleteConnectionSecurityRule (Opnum 14)](#section_b470f66f926247479be719f8aa92a611) 143

[RRPC\_FWDeleteCryptoSet (Opnum 24)](#section_6c5b4c64545c4b108208414de327ead9) 154

[RRPC\_FWDeleteFirewallRule (Opnum 7)](#section_e1ad0b524218420bb982d5035f6c8da3) 136

[RRPC\_FWDeleteMainModeRule (Opnum 34)](#section_7523b410086443278bb002d00ffa6d2b) 163

[RRPC\_FWDeletePhase1SAs (Opnum 29)](#section_0666c872b08f4b82a263e1935b763141) 159

[RRPC\_FWDeletePhase2SAs (Opnum 30)](#section_1b6a54e35aba4bc49b9629b5514d585f) 160

[RRPC\_FWEnumAdapters (Opnum 43)](#section_c5c35afeb29e433bb7bcca0ffa9adcd4) 171

[RRPC\_FWEnumAuthenticationSets (Opnum 21)](#section_2dc9c31538e842d69c88927dea828314) 150

[RRPC\_FWEnumAuthenticationSets2\_10 (Opnum 54)](#section_a788ee5f53964fd19d807e31d639b770) 184

[RRPC\_FWEnumAuthenticationSets2\_20 (Opnum 64)](#section_be14dfc72ce64197af85575d8e96873d) 194

[RRPC\_FWEnumConnectionSecurityRules (Opnum 16)](#section_f0f85db355104b73a4cb76bd440eaa06) 145

[RRPC\_FWEnumConnectionSecurityRules2\_10 (Opnum 51)](#section_8e2bce0a21e4412897473b249c0b0091) 181

[RRPC\_FWEnumConnectionSecurityRules2\_20 (Opnum 60)](#section_be8c8ed3570e45c0b063bffeec096ede) 190

[RRPC\_FWEnumCryptoSets (Opnum 26)](#section_63e911eafbca409689a95d84c37daa56) 156

[RRPC\_FWEnumCryptoSets2\_10 (Opnum 57)](#section_d036149f27ed45d1a55df7881b50b3f1) 187

[RRPC\_FWEnumFirewallRules (Opnum 9)](#section_469f8466bb5c4af097a8335f648c70d7) 137

[RRPC\_FWEnumFirewallRules2\_10 (Opnum 48)](#section_77e5217a46d8440ab330679a0939c9ae) 177

[RRPC\_FWEnumFirewallRules2\_20 (Opnum 68)](#section_647f1dac0be1415896c906b1e03c6bdb) 198

[RRPC\_FWEnumFirewallRules2\_24 (Opnum 72)](#section_ee81af2bc01c4cf3812ebab6c89bd6ed) 203

[RRPC\_FWEnumFirewallRules2\_25 (Opnum 76)](#section_17e6234cbd8e48439a2f884801ceddc9) 207

[RRPC\_FWEnumFirewallRules2\_26 (Opnum 80)](#section_cd34e45d4a254c928d34a07f3082c28e) 211

[RRPC\_FWEnumFirewallRules2\_27 (Opnum 84)](#section_539b1f29aba74d75b034b5f9db68bf81) 215

[RRPC\_FWEnumMainModeRules (Opnum 36)](#section_70ad911aa488474289549b9d60af3075) 165

[RRPC\_FWEnumNetworks (Opnum 42)](#section_b3f615b50a51424cab3c15b3fb350ee1) 171

[RRPC\_FWEnumPhase1SAs (Opnum 27)](#section_c97e073507544e688c82fb3b4275d085) 157

[RRPC\_FWEnumPhase2SAs (Opnum 28)](#section_2c4f14fc0fee41f0b1f3e0082c929995) 158

[RRPC\_FWEnumProducts (Opnum 31)](#section_b7a0c4f8f7a948f1b542a2937e269ca3) 161

[RRPC\_FWGetConfig (Opnum 10)](#section_debe35e3cba149ceb331374d4e88d0ae) 138

[RRPC\_FWGetConfig2\_10 (Opnum 45)](#section_3de45ae3d1c741a88061c435cf72e480) 174

[RRPC\_FWGetGlobalConfig (Opnum 3)](#section_10c07a562a4f4f068510c6f3bfe5ca4a) 131

[RRPC\_FWGetGlobalConfig2\_10 (Opnum 44)](#section_e479f36dc2254c1cbb67d46c43dc7c7e) 172

[RRPC\_FWOpenPolicyStore (Opnum 0)](#section_2157e39a1bf04e0cabae0811fd918b11) 129

[RRPC\_FWQueryAuthenticationSets (Opnum 40)](#section_1b4f80300bb24a1ca3e57151d5f9616b) 169

[RRPC\_FWQueryAuthenticationSets2\_20 (Opnum 65)](#section_71836286fe1346a6a81ffed593a0fb77) 195

[RRPC\_FWQueryConnectionSecurityRules (Opnum 38)](#section_b092c4bcf6d54f44b792599839c9db2f) 167

[RRPC\_FWQueryConnectionSecurityRules2\_20 (Opnum 61)](#section_ce8fd2ce757a4ffa869d7ba9056bf68d) 191

[RRPC\_FWQueryCryptoSets (Opnum 41)](#section_3dc9e91dd0ee4e1ca30d70c89052de84) 170

[RRPC\_FWQueryFirewallRules (Opnum 37)](#section_3422b6ac5b04456dbc4ffd97e1384c3a) 166

[RRPC\_FWQueryFirewallRules2\_20 (Opnum 69)](#section_7d1c340631a3440fa6d47232dd8bb36e) 199

[RRPC\_FWQueryFirewallRules2\_24 (Opnum 73)](#section_0c82d80e54e34daf9cff4a92a54eaa84) 204

[RRPC\_FWQueryFirewallRules2\_25 (Opnum 77)](#section_d6e826512e294f9999f13e2bd7cb93bd) 208

[RRPC\_FWQueryFirewallRules2\_26 (Opnum 81)](#section_cf0aae4fbb2a42929bffd10b0b89f36e) 212

[RRPC\_FWQueryFirewallRules2\_27 (Opnum 85)](#section_5d9f60b806f94a0585f9a54383265679) 216

[RRPC\_FWQueryMainModeRules (Opnum 39)](#section_dc7a4aa81cb34c1f8506e6ca581d4434) 168

[RRPC\_FWRestoreDefaults (Opnum 2)](#section_764123cab72743229fe1de1fd859fac1) 130

[RRPC\_FWSetAuthenticationSet (Opnum 18)](#section_5a5b0f3ad8ac45608a6a485ece2cbb6c) 147

[RRPC\_FWSetAuthenticationSet2\_10 (Opnum 53)](#section_dfb2e973f6c64b57908afa094d90ff83) 183

[RRPC\_FWSetAuthenticationSet2\_20 (Opnum 63)](#section_4023224c13d14df4995761b0a62937ed) 193

[RRPC\_FWSetConfig (Opnum 11)](#section_8621c5249ccc495c9c80a0830282ccf8) 140

[RRPC\_FWSetConnectionSecurityRule (Opnum 13)](#section_c66ad9a72ee44d82a62e73ce5599c05c) 142

[RRPC\_FWSetConnectionSecurityRule2\_10 (Opnum 50)](#section_4229bd7163c946a1a97e3c061604d717) 180

[RRPC\_FWSetConnectionSecurityRule2\_20 (Opnum 59)](#section_b255739072f24d1da7db790853f2df36) 189

[RRPC\_FWSetCryptoSet (Opnum 23)](#section_423d057609384c818c9beacafc69504c) 152

[RRPC\_FWSetCryptoSet2\_10 (Opnum 56)](#section_0b68e0c61c2844eb93a06406746d8814) 186

[RRPC\_FWSetFirewallRule (Opnum 6)](#section_676f90d026b44ed1a5eb1f88db8676b4) 135

[RRPC\_FWSetFirewallRule2\_10 (Opnum 47)](#section_27a5eb1dc5b94ccaa50263e75df80917) 176

[RRPC\_FWSetFirewallRule2\_20 (Opnum 67)](#section_cb987e3d916f4902aaba4b99df2e3af3) 197

[RRPC\_FWSetFirewallRule2\_24 (Opnum 71)](#section_f6e7c7a0ffc74f72a20a0b57e784bd09) 202

[RRPC\_FWSetFirewallRule2\_25 (Opnum 75)](#section_7d89953750424032bbdc50cec6656c2c) 206

[RRPC\_FWSetFirewallRule2\_26 (Opnum 79)](#section_d8b1914636d14a479d31bf0194f2b9da) 210

[RRPC\_FWSetFirewallRule2\_27 (Opnum 83)](#section_0ab23c3cdfd1452d9026115ac71dc9ae) 214

[RRPC\_FWSetGlobalConfig (Opnum 4)](#section_45676b5c298b413b8fac0770c3def938) 132

[RRPC\_FWSetMainModeRule (Opnum 33)](#section_268116a5af2b407b850c1380bece5169) 162

N

[Normative references](#section_cc212a1f979f4a39938bb3488a157c4b) 12

O

[Opening a policy store example](#section_230d1ae7b42e4d9cb997b1463aaa0ded) 221

[Opening policy store example](#section_230d1ae7b42e4d9cb997b1463aaa0ded) 221

[Overview](#section_3e3c5b45430c43c684421bba3853f9d9) 13

[Overview (synopsis)](#section_3e3c5b45430c43c684421bba3853f9d9) 13

P

[Parameter index - security](#section_ea9a532ff18a4d32a65413182cb99338) 225

[Parameters - security index](#section_ea9a532ff18a4d32a65413182cb99338) 225

[PFW\_ADAPTER](#section_24178f7ad5f749bda7ee69fe2b02ff4f) 64

[PFW\_ADDRESSES](#section_67b07a7f8748487e8fd8c75fe3b307e6) 28

[PFW\_AUTH\_INFO](#section_cd08ebed295d4c7786c64ba87aa4df00) 97

[PFW\_AUTH\_SET](#section_bbec7ab70a1d49a49c8ea4784bbe6f16) 83

[PFW\_AUTH\_SET2\_10](#section_c1acee0a747a4482b1590c96ca57fe29) 80

[PFW\_AUTH\_SUITE](#section_447c3a4c9543495dab7db7ca639712ef) 79

[PFW\_AUTH\_SUITE2\_10](#section_bae6ee78bedc4a3a95458920c2a93cda) 78

[PFW\_BYTE\_BLOB](#section_8e7e10f0590043618403d8833934c491) 95

[PFW\_CERT\_CRITERIA](#section_8f4e2f51826f43c794055395b824bd41) 75

[PFW\_CERT\_INFO](#section_dfc16343eb514bbab890df5a58839b2b) 96

[PFW\_COOKIE\_PAIR](#section_1974c8d425694bcea0cbbc692963fca1) 96

[PFW\_CRYPTO\_SET](#section_a468fe9e113b4155a63d0db3aac12619) 92

[PFW\_CS\_RULE](#section_0d0641105f2e4b68aa63032c6cd5e4c6) 70

[PFW\_CS\_RULE2\_0](#section_18277e4c628d44239fe3ba98601ba50d) 69

[PFW\_CS\_RULE2\_10](#section_8f1281f0bb53491fa552742a1a922137) 68

[PFW\_ENDPOINTS](#section_2318781e64bc475d997502cc9d56ef0b) 97

[PFW\_ICMP\_TYPE\_CODE](#section_6feb51f6495e4396b86c0dbfc5650dff) 25

[PFW\_ICMP\_TYPE\_CODE\_LIST](#section_efce15b5aabb4273abce078aca9c6427) 25

[PFW\_INTERFACE\_LUIDS](#section_ea420d0f03ed48e5b786621db56419d5) 26

[PFW\_IPV4\_ADDRESS\_RANGE](#section_f11b849900a043d8bf3a6b596ec0d55c) 22

[PFW\_IPV4\_RANGE\_LIST](#section_de836948bf68424d906d406723bd0deb) 22

[PFW\_IPV4\_SUBNET](#section_ed0a65ecd3b6461ca4caecf08ed18025) 21

[PFW\_IPV4\_SUBNET\_LIST](#section_a230f0e48005498389e3e417b3354ae5) 21

[PFW\_IPV6\_ADDRESS\_RANGE](#section_fbf349cc8455490db97d951c63084861) 22

[PFW\_IPV6\_RANGE\_LIST](#section_062373f2c3dc49a98b3f960d2a88332d) 23

[PFW\_IPV6\_SUBNET](#section_ccb501eb66204b49befd2b59828a28c8) 21

[PFW\_IPV6\_SUBNET\_LIST](#section_a8d05086bc48453eaebfaa11fb95b9bd) 21

[PFW\_MM\_RULE](#section_4b3fc163fede434c90cb557992caa5da) 101

[PFW\_NETWORK](#section_682c407a18024e9eb96a8c74315e5746) 64

[PFW\_NETWORK\_NAMES](#section_1e72af2f869f484ab1478371ce830a0c) 111

[PFW\_OBJECT\_METADATA](#section_cecfb58e2c794b7db92c7666fa06d559) 45

[PFW\_OS\_PLATFORM](#section_aac2be8cf4ab4b0a995774462137fd8a) 46

[PFW\_OS\_PLATFORM\_LIST](#section_dae582a2831e4e6a8dc923be7358fefd) 47

[PFW\_PHASE1\_CRYPTO\_SUITE](#section_cd273bb434214831a979347630eb3b16) 89

[PFW\_PHASE1\_SA\_DETAILS](#section_218b010dea7f4b14b728aa27dc418236) 98

[PFW\_PHASE2\_CRYPTO\_SUITE](#section_99a997b6de524648977ce9efd6058254) 90

[PFW\_PHASE2\_SA\_DETAILS](#section_e6091d0ae57846e08656113d1e1e360d) 99

[PFW\_PORT\_RANGE](#section_3122d4b1950043af900a27fe64688623) 23

[PFW\_PORT\_RANGE\_LIST](#section_5d0e412713fe46b38685f7784941c5b4) 23

[PFW\_PORTS](#section_df16981d5d474a25aed73d03cf33998a) 25

[PFW\_PRODUCT](#section_969d38731e424dc48559f5c0c9caa965) 65

[PFW\_QUERY](#section_51b69145317a4368a6e57159e946ff14) 107

[PFW\_QUERY\_CONDITION](#section_fa19141ffc3c45159555b6db62c475aa) 106

[PFW\_QUERY\_CONDITIONS](#section_98765a8c9a0c471bb2ee56c931ea5604) 107

[PFW\_RULE](#section_8c008258166d46d49090f2ffaa01be4b) 52

[PFW\_RULE2\_0](#section_af151922b0924e5c9b694dca6b6d2ffc) 51

[Preconditions](#section_b61949b0e2634f38a6beb586a4a075fc) 16

[Prerequisites](#section_b61949b0e2634f38a6beb586a4a075fc) 16

[Product behavior](#section_1da2ee70a6ae4f76b08ffdc25c77d8a0) 285

Protocol Details

[overview](#section_e80a2f89377e4429b18e74ead529938e) 117

R

[References](#section_fa2a61575c4945229372c0aef8225f50) 12

[informative](#section_cbdceb4d03bb4b918c70f2ec74fad0bc) 12

[normative](#section_cc212a1f979f4a39938bb3488a157c4b) 12

[Relationship to other protocols](#section_fdbf2915c711482492d2ab572ab49a52) 15

[RRPC\_FWAddAuthenticationSet (Opnum 17) method](#section_c1dbe17a32cf4910843b7d2690e9087a) 146

[RRPC\_FWAddAuthenticationSet method](#section_c1dbe17a32cf4910843b7d2690e9087a) 146

[RRPC\_FWAddAuthenticationSet2\_10 (Opnum 52) method](#section_25bb2543bda546e5ba7afa6c7a356ba4) 182

[RRPC\_FWAddAuthenticationSet2\_10 method](#section_25bb2543bda546e5ba7afa6c7a356ba4) 182

[RRPC\_FWAddAuthenticationSet2\_20 (Opnum 62) method](#section_b098dcdb7058470dba9df042dec221c0) 192

[RRPC\_FWAddAuthenticationSet2\_20 method](#section_b098dcdb7058470dba9df042dec221c0) 192

[RRPC\_FWAddConnectionSecurityRule (Opnum 12) method](#section_4e31fcb5b70148b58a45d70bf85281ab) 141

[RRPC\_FWAddConnectionSecurityRule method](#section_4e31fcb5b70148b58a45d70bf85281ab) 141

[RRPC\_FWAddConnectionSecurityRule2\_10 (Opnum 49) method](#section_b7a0d5cca02f46078681e8b5589d3451) 179

[RRPC\_FWAddConnectionSecurityRule2\_10 method](#section_b7a0d5cca02f46078681e8b5589d3451) 179

[RRPC\_FWAddConnectionSecurityRule2\_20 (Opnum 58) method](#section_222c230acb4c41b2ac44a770cd71d66c) 188

RRPC\_FWAddConnectionSecurityRule2\_20 method ([section 3.1.4.59](#section_222c230acb4c41b2ac44a770cd71d66c) 188, [section 3.1.4.68](#section_cb987e3d916f4902aaba4b99df2e3af3) 197)

[RRPC\_FWAddCryptoSet (Opnum 22) method](#section_0399561877894a00828cd8c497b643e3) 151

[RRPC\_FWAddCryptoSet method](#section_0399561877894a00828cd8c497b643e3) 151

[RRPC\_FWAddCryptoSet2\_10 (Opnum 55) method](#section_604883c63e674140bec0ea68adf4eb53) 185

[RRPC\_FWAddCryptoSet2\_10 method](#section_604883c63e674140bec0ea68adf4eb53) 185

[RRPC\_FWAddFirewallRule (Opnum 5) method](#section_4d7baada78fd469a9c292885e5d3b5d0) 134

[RRPC\_FWAddFirewallRule method](#section_4d7baada78fd469a9c292885e5d3b5d0) 134

[RRPC\_FWAddFirewallRule2\_10 (Opnum 46) method](#section_5ab52f049e9d426b854560e2367fdffd) 175

[RRPC\_FWAddFirewallRule2\_10 method](#section_5ab52f049e9d426b854560e2367fdffd) 175

[RRPC\_FWAddFirewallRule2\_20 (Opnum 66) method](#section_d637fee2b28d4cea8005303347de94f7) 196

[RRPC\_FWAddFirewallRule2\_20 method](#section_d637fee2b28d4cea8005303347de94f7) 196

[RRPC\_FWAddFirewallRule2\_24 (Opnum 70) method](#section_778f196ee6f347bfa3a6f6f5f7660b21) 200

[RRPC\_FWAddFirewallRule2\_24 method](#section_778f196ee6f347bfa3a6f6f5f7660b21) 200

[RRPC\_FWAddFirewallRule2\_25 (Opnum 74) method](#section_fb298ceac5fe4609b71ce0221dde97c1) 205

[RRPC\_FWAddFirewallRule2\_26 (Opnum 78) method](#section_944db4f78e434b1b9060d9bd4ef8e7d1) 209

[RRPC\_FWAddFirewallRule2\_27 (Opnum 82) method](#section_980933b01319416a94ee374703462728) 213

[RRPC\_FWAddMainModeRule (Opnum 32) method](#section_40973e9f20c84e7b8f96f2664a622f03) 162

[RRPC\_FWAddMainModeRule method](#section_40973e9f20c84e7b8f96f2664a622f03) 162

[RRPC\_FWClosePolicyStore (Opnum 1) method](#section_203dc96f2e74449aae18aaae0689c092) 130

[RRPC\_FWClosePolicyStore method](#section_203dc96f2e74449aae18aaae0689c092) 130

[RRPC\_FWDeleteAllAuthenticationSets (Opnum 20) method](#section_7d8f82f7d9284ebc8afc576751e2e8d2) 149

[RRPC\_FWDeleteAllAuthenticationSets method](#section_7d8f82f7d9284ebc8afc576751e2e8d2) 149

[RRPC\_FWDeleteAllConnectionSecurityRules (Opnum 15) method](#section_ee4f85d9c07449658508c8ec96ae3b77) 144

[RRPC\_FWDeleteAllConnectionSecurityRules method](#section_ee4f85d9c07449658508c8ec96ae3b77) 144

[RRPC\_FWDeleteAllCryptoSets (Opnum 25) method](#section_67cf747ee0054630932928887df1e09d) 155

[RRPC\_FWDeleteAllCryptoSets method](#section_67cf747ee0054630932928887df1e09d) 155

[RRPC\_FWDeleteAllFirewallRules (Opnum 8) method](#section_ca2da2f2b8ef4981abbafed5c713990d) 136

[RRPC\_FWDeleteAllFirewallRules method](#section_ca2da2f2b8ef4981abbafed5c713990d) 136

[RRPC\_FWDeleteAllMainModeRules (Opnum 35) method](#section_d47dfd57c6b9448eb5a0127d3f87cd38) 164

[RRPC\_FWDeleteAllMainModeRules method](#section_d47dfd57c6b9448eb5a0127d3f87cd38) 164

[RRPC\_FWDeleteAuthenticationSet (Opnum 19) method](#section_b143f0f85f274595b5310197fc491d0b) 148

[RRPC\_FWDeleteAuthenticationSet method](#section_b143f0f85f274595b5310197fc491d0b) 148

[RRPC\_FWDeleteConnectionSecurityRule (Opnum 14) method](#section_b470f66f926247479be719f8aa92a611) 143

[RRPC\_FWDeleteConnectionSecurityRule method](#section_b470f66f926247479be719f8aa92a611) 143

[RRPC\_FWDeleteCryptoSet (Opnum 24) method](#section_6c5b4c64545c4b108208414de327ead9) 154

[RRPC\_FWDeleteCryptoSet method](#section_6c5b4c64545c4b108208414de327ead9) 154

[RRPC\_FWDeleteFirewallRule (Opnum 7) method](#section_e1ad0b524218420bb982d5035f6c8da3) 136

[RRPC\_FWDeleteFirewallRule method](#section_e1ad0b524218420bb982d5035f6c8da3) 136

[RRPC\_FWDeleteMainModeRule (Opnum 34) method](#section_7523b410086443278bb002d00ffa6d2b) 163

[RRPC\_FWDeleteMainModeRule method](#section_7523b410086443278bb002d00ffa6d2b) 163

[RRPC\_FWDeletePhase1SAs (Opnum 29) method](#section_0666c872b08f4b82a263e1935b763141) 159

[RRPC\_FWDeletePhase1SAs method](#section_0666c872b08f4b82a263e1935b763141) 159

[RRPC\_FWDeletePhase2SAs (Opnum 30) method](#section_1b6a54e35aba4bc49b9629b5514d585f) 160

[RRPC\_FWDeletePhase2SAs method](#section_1b6a54e35aba4bc49b9629b5514d585f) 160

[RRPC\_FWEnumAdapters (Opnum 43) method](#section_c5c35afeb29e433bb7bcca0ffa9adcd4) 171

[RRPC\_FWEnumAdapters method](#section_c5c35afeb29e433bb7bcca0ffa9adcd4) 171

[RRPC\_FWEnumAuthenticationSets (Opnum 21) method](#section_2dc9c31538e842d69c88927dea828314) 150

[RRPC\_FWEnumAuthenticationSets method](#section_2dc9c31538e842d69c88927dea828314) 150

[RRPC\_FWEnumAuthenticationSets2\_10 (Opnum 54) method](#section_a788ee5f53964fd19d807e31d639b770) 184

[RRPC\_FWEnumAuthenticationSets2\_10 method](#section_a788ee5f53964fd19d807e31d639b770) 184

[RRPC\_FWEnumAuthenticationSets2\_20 (Opnum 64) method](#section_be14dfc72ce64197af85575d8e96873d) 194

[RRPC\_FWEnumAuthenticationSets2\_20 method](#section_be14dfc72ce64197af85575d8e96873d) 194

[RRPC\_FWEnumConnectionSecurityRules (Opnum 16) method](#section_f0f85db355104b73a4cb76bd440eaa06) 145

[RRPC\_FWEnumConnectionSecurityRules method](#section_f0f85db355104b73a4cb76bd440eaa06) 145

[RRPC\_FWEnumConnectionSecurityRules2\_10 (Opnum 51) method](#section_8e2bce0a21e4412897473b249c0b0091) 181

[RRPC\_FWEnumConnectionSecurityRules2\_10 method](#section_8e2bce0a21e4412897473b249c0b0091) 181

[RRPC\_FWEnumConnectionSecurityRules2\_20 (Opnum 60) method](#section_be8c8ed3570e45c0b063bffeec096ede) 190

[RRPC\_FWEnumConnectionSecurityRules2\_20 method](#section_be8c8ed3570e45c0b063bffeec096ede) 190

[RRPC\_FWEnumCryptoSets (Opnum 26) method](#section_63e911eafbca409689a95d84c37daa56) 156

[RRPC\_FWEnumCryptoSets method](#section_63e911eafbca409689a95d84c37daa56) 156

[RRPC\_FWEnumCryptoSets2\_10 (Opnum 57) method](#section_d036149f27ed45d1a55df7881b50b3f1) 187

[RRPC\_FWEnumCryptoSets2\_10 method](#section_d036149f27ed45d1a55df7881b50b3f1) 187

[RRPC\_FWEnumFirewallRules (Opnum 9) method](#section_469f8466bb5c4af097a8335f648c70d7) 137

[RRPC\_FWEnumFirewallRules method](#section_469f8466bb5c4af097a8335f648c70d7) 137

[RRPC\_FWEnumFirewallRules2\_10 (Opnum 48) method](#section_77e5217a46d8440ab330679a0939c9ae) 177

[RRPC\_FWEnumFirewallRules2\_10 method](#section_77e5217a46d8440ab330679a0939c9ae) 177

[RRPC\_FWEnumFirewallRules2\_20 (Opnum 68) method](#section_647f1dac0be1415896c906b1e03c6bdb) 198

[RRPC\_FWEnumFirewallRules2\_20 method](#section_647f1dac0be1415896c906b1e03c6bdb) 198

[RRPC\_FWEnumFirewallRules2\_24 (Opnum 72) method](#section_ee81af2bc01c4cf3812ebab6c89bd6ed) 203

[RRPC\_FWEnumFirewallRules2\_24 method](#section_ee81af2bc01c4cf3812ebab6c89bd6ed) 203

[RRPC\_FWEnumFirewallRules2\_25 (Opnum 76) method](#section_17e6234cbd8e48439a2f884801ceddc9) 207

[RRPC\_FWEnumFirewallRules2\_26 (Opnum 80) method](#section_cd34e45d4a254c928d34a07f3082c28e) 211

[RRPC\_FWEnumFirewallRules2\_27 (Opnum 84) method](#section_539b1f29aba74d75b034b5f9db68bf81) 215

[RRPC\_FWEnumMainModeRules (Opnum 36) method](#section_70ad911aa488474289549b9d60af3075) 165

[RRPC\_FWEnumMainModeRules method](#section_70ad911aa488474289549b9d60af3075) 165

[RRPC\_FWEnumNetworks (Opnum 42) method](#section_b3f615b50a51424cab3c15b3fb350ee1) 171

[RRPC\_FWEnumNetworks method](#section_b3f615b50a51424cab3c15b3fb350ee1) 171

[RRPC\_FWEnumPhase1SAs (Opnum 27) method](#section_c97e073507544e688c82fb3b4275d085) 157

[RRPC\_FWEnumPhase1SAs method](#section_c97e073507544e688c82fb3b4275d085) 157

[RRPC\_FWEnumPhase2SAs (Opnum 28) method](#section_2c4f14fc0fee41f0b1f3e0082c929995) 158

[RRPC\_FWEnumPhase2SAs method](#section_2c4f14fc0fee41f0b1f3e0082c929995) 158

[RRPC\_FWEnumProducts (Opnum 31) method](#section_b7a0c4f8f7a948f1b542a2937e269ca3) 161

[RRPC\_FWEnumProducts method](#section_b7a0c4f8f7a948f1b542a2937e269ca3) 161

[RRPC\_FWGetConfig (Opnum 10) method](#section_debe35e3cba149ceb331374d4e88d0ae) 138

[RRPC\_FWGetConfig method](#section_debe35e3cba149ceb331374d4e88d0ae) 138

[RRPC\_FWGetConfig2\_10 (Opnum 45) method](#section_3de45ae3d1c741a88061c435cf72e480) 174

[RRPC\_FWGetConfig2\_10 method](#section_3de45ae3d1c741a88061c435cf72e480) 174

[RRPC\_FWGetGlobalConfig (Opnum 3) method](#section_10c07a562a4f4f068510c6f3bfe5ca4a) 131

[RRPC\_FWGetGlobalConfig method](#section_10c07a562a4f4f068510c6f3bfe5ca4a) 131

[RRPC\_FWGetGlobalConfig2\_10 (Opnum 44) method](#section_e479f36dc2254c1cbb67d46c43dc7c7e) 172

[RRPC\_FWGetGlobalConfig2\_10 method](#section_e479f36dc2254c1cbb67d46c43dc7c7e) 172

[RRPC\_FWModifyConnectionSecurityRule2\_20 method](#section_b255739072f24d1da7db790853f2df36) 189

[RRPC\_FWOpenPolicyStore (Opnum 0) method](#section_2157e39a1bf04e0cabae0811fd918b11) 129

[RRPC\_FWOpenPolicyStore method](#section_2157e39a1bf04e0cabae0811fd918b11) 129

[RRPC\_FWQueryAuthenticationSets (Opnum 40) method](#section_1b4f80300bb24a1ca3e57151d5f9616b) 169

[RRPC\_FWQueryAuthenticationSets method](#section_1b4f80300bb24a1ca3e57151d5f9616b) 169

[RRPC\_FWQueryAuthenticationSets2\_20 (Opnum 65) method](#section_71836286fe1346a6a81ffed593a0fb77) 195

[RRPC\_FWQueryAuthenticationSets2\_20 method](#section_71836286fe1346a6a81ffed593a0fb77) 195

[RRPC\_FWQueryConnectionSecurityRules (Opnum 38) method](#section_b092c4bcf6d54f44b792599839c9db2f) 167

[RRPC\_FWQueryConnectionSecurityRules method](#section_b092c4bcf6d54f44b792599839c9db2f) 167

[RRPC\_FWQueryConnectionSecurityRules2\_20 (Opnum 61) method](#section_ce8fd2ce757a4ffa869d7ba9056bf68d) 191

[RRPC\_FWQueryConnectionSecurityRules2\_20 method](#section_ce8fd2ce757a4ffa869d7ba9056bf68d) 191

[RRPC\_FWQueryCryptoSets (Opnum 41) method](#section_3dc9e91dd0ee4e1ca30d70c89052de84) 170

[RRPC\_FWQueryCryptoSets method](#section_3dc9e91dd0ee4e1ca30d70c89052de84) 170

[RRPC\_FWQueryFirewallRules (Opnum 37) method](#section_3422b6ac5b04456dbc4ffd97e1384c3a) 166

[RRPC\_FWQueryFirewallRules method](#section_3422b6ac5b04456dbc4ffd97e1384c3a) 166

[RRPC\_FWQueryFirewallRules2\_20 (Opnum 69) method](#section_7d1c340631a3440fa6d47232dd8bb36e) 199

[RRPC\_FWQueryFirewallRules2\_20 method](#section_7d1c340631a3440fa6d47232dd8bb36e) 199

[RRPC\_FWQueryFirewallRules2\_24 (Opnum 73) method](#section_0c82d80e54e34daf9cff4a92a54eaa84) 204

[RRPC\_FWQueryFirewallRules2\_24 method](#section_0c82d80e54e34daf9cff4a92a54eaa84) 204

[RRPC\_FWQueryFirewallRules2\_25 (Opnum 77) method](#section_d6e826512e294f9999f13e2bd7cb93bd) 208

[RRPC\_FWQueryFirewallRules2\_26 (Opnum 81) method](#section_cf0aae4fbb2a42929bffd10b0b89f36e) 212

[RRPC\_FWQueryFirewallRules2\_27 (Opnum 85) method](#section_5d9f60b806f94a0585f9a54383265679) 216

[RRPC\_FWQueryMainModeRules (Opnum 39) method](#section_dc7a4aa81cb34c1f8506e6ca581d4434) 168

[RRPC\_FWQueryMainModeRules method](#section_dc7a4aa81cb34c1f8506e6ca581d4434) 168

[RRPC\_FWRestoreDefaults (Opnum 2) method](#section_764123cab72743229fe1de1fd859fac1) 130

[RRPC\_FWRestoreDefaults method](#section_764123cab72743229fe1de1fd859fac1) 130

[RRPC\_FWSetAuthenticationSet (Opnum 18) method](#section_5a5b0f3ad8ac45608a6a485ece2cbb6c) 147

[RRPC\_FWSetAuthenticationSet method](#section_5a5b0f3ad8ac45608a6a485ece2cbb6c) 147

[RRPC\_FWSetAuthenticationSet2\_10 (Opnum 53) method](#section_dfb2e973f6c64b57908afa094d90ff83) 183

[RRPC\_FWSetAuthenticationSet2\_10 method](#section_dfb2e973f6c64b57908afa094d90ff83) 183

[RRPC\_FWSetAuthenticationSet2\_20 (Opnum 63) method](#section_4023224c13d14df4995761b0a62937ed) 193

[RRPC\_FWSetAuthenticationSet2\_20 method](#section_4023224c13d14df4995761b0a62937ed) 193

[RRPC\_FWSetConfig (Opnum 11) method](#section_8621c5249ccc495c9c80a0830282ccf8) 140

[RRPC\_FWSetConfig method](#section_8621c5249ccc495c9c80a0830282ccf8) 140

[RRPC\_FWSetConnectionSecurityRule (Opnum 13) method](#section_c66ad9a72ee44d82a62e73ce5599c05c) 142

[RRPC\_FWSetConnectionSecurityRule method](#section_c66ad9a72ee44d82a62e73ce5599c05c) 142

[RRPC\_FWSetConnectionSecurityRule2\_10 (Opnum 50) method](#section_4229bd7163c946a1a97e3c061604d717) 180

[RRPC\_FWSetConnectionSecurityRule2\_10 method](#section_4229bd7163c946a1a97e3c061604d717) 180

[RRPC\_FWSetConnectionSecurityRule2\_20 (Opnum 59) method](#section_b255739072f24d1da7db790853f2df36) 189

[RRPC\_FWSetCryptoSet (Opnum 23) method](#section_423d057609384c818c9beacafc69504c) 152

[RRPC\_FWSetCryptoSet method](#section_423d057609384c818c9beacafc69504c) 152

[RRPC\_FWSetCryptoSet2\_10 (Opnum 56) method](#section_0b68e0c61c2844eb93a06406746d8814) 186

[RRPC\_FWSetCryptoSet2\_10 method](#section_0b68e0c61c2844eb93a06406746d8814) 186

[RRPC\_FWSetFirewallRule (Opnum 6) method](#section_676f90d026b44ed1a5eb1f88db8676b4) 135

[RRPC\_FWSetFirewallRule method](#section_676f90d026b44ed1a5eb1f88db8676b4) 135

[RRPC\_FWSetFirewallRule2\_10 (Opnum 47) method](#section_27a5eb1dc5b94ccaa50263e75df80917) 176

[RRPC\_FWSetFirewallRule2\_10 method](#section_27a5eb1dc5b94ccaa50263e75df80917) 176

[RRPC\_FWSetFirewallRule2\_20 (Opnum 67) method](#section_cb987e3d916f4902aaba4b99df2e3af3) 197

[RRPC\_FWSetFirewallRule2\_24 (Opnum 71) method](#section_f6e7c7a0ffc74f72a20a0b57e784bd09) 202

[RRPC\_FWSetFirewallRule2\_24 method](#section_f6e7c7a0ffc74f72a20a0b57e784bd09) 202

[RRPC\_FWSetFirewallRule2\_25 (Opnum 75) method](#section_7d89953750424032bbdc50cec6656c2c) 206

[RRPC\_FWSetFirewallRule2\_26 (Opnum 79) method](#section_d8b1914636d14a479d31bf0194f2b9da) 210

[RRPC\_FWSetFirewallRule2\_27 (Opnum 83) method](#section_0ab23c3cdfd1452d9026115ac71dc9ae) 214

[RRPC\_FWSetGlobalConfig (Opnum 4) method](#section_45676b5c298b413b8fac0770c3def938) 132

[RRPC\_FWSetGlobalConfig method](#section_45676b5c298b413b8fac0770c3def938) 132

[RRPC\_FWSetMainModeRule (Opnum 33) method](#section_268116a5af2b407b850c1380bece5169) 162

[RRPC\_FWSetMainModeRule method](#section_268116a5af2b407b850c1380bece5169) 162

S

Security

[implementer considerations](#section_41e32b09744241898b7717ded87b4751) 225

[parameter index](#section_ea9a532ff18a4d32a65413182cb99338) 225

Sequencing rules

[client](#section_0bf6ef331e1548d081172f79d2f21e5c) 220

[server](#section_b0c9335283714d63abcab0cc8dbcc3d7) 121

Server

[abstract data model](#section_43507d538955416db913dfb27dc76b17) 117

[initialization](#section_e8924ac5aa4a41d1bf654f46b3d399aa) 120

[local events](#section_6e5c3ec0c748485687c99d6b35ab7280) 217

[message processing](#section_b0c9335283714d63abcab0cc8dbcc3d7) 121

[RRPC\_FWAddAuthenticationSet (Opnum 17) method](#section_c1dbe17a32cf4910843b7d2690e9087a) 146

[RRPC\_FWAddAuthenticationSet2\_10 (Opnum 52) method](#section_25bb2543bda546e5ba7afa6c7a356ba4) 182

[RRPC\_FWAddAuthenticationSet2\_20 (Opnum 62) method](#section_b098dcdb7058470dba9df042dec221c0) 192

[RRPC\_FWAddConnectionSecurityRule (Opnum 12) method](#section_4e31fcb5b70148b58a45d70bf85281ab) 141

[RRPC\_FWAddConnectionSecurityRule2\_10 (Opnum 49) method](#section_b7a0d5cca02f46078681e8b5589d3451) 179

[RRPC\_FWAddConnectionSecurityRule2\_20 (Opnum 58) method](#section_222c230acb4c41b2ac44a770cd71d66c) 188

[RRPC\_FWAddCryptoSet (Opnum 22) method](#section_0399561877894a00828cd8c497b643e3) 151

[RRPC\_FWAddCryptoSet2\_10 (Opnum 55) method](#section_604883c63e674140bec0ea68adf4eb53) 185

[RRPC\_FWAddFirewallRule (Opnum 5) method](#section_4d7baada78fd469a9c292885e5d3b5d0) 134

[RRPC\_FWAddFirewallRule2\_10 (Opnum 46) method](#section_5ab52f049e9d426b854560e2367fdffd) 175

[RRPC\_FWAddFirewallRule2\_20 (Opnum 66) method](#section_d637fee2b28d4cea8005303347de94f7) 196

[RRPC\_FWAddFirewallRule2\_24 (Opnum 70) method](#section_778f196ee6f347bfa3a6f6f5f7660b21) 200

[RRPC\_FWAddFirewallRule2\_25 (Opnum 74) method](#section_fb298ceac5fe4609b71ce0221dde97c1) 205

[RRPC\_FWAddFirewallRule2\_26 (Opnum 78) method](#section_944db4f78e434b1b9060d9bd4ef8e7d1) 209

[RRPC\_FWAddFirewallRule2\_27 (Opnum 82) method](#section_980933b01319416a94ee374703462728) 213

[RRPC\_FWAddMainModeRule (Opnum 32) method](#section_40973e9f20c84e7b8f96f2664a622f03) 162

[RRPC\_FWClosePolicyStore (Opnum 1) method](#section_203dc96f2e74449aae18aaae0689c092) 130

[RRPC\_FWDeleteAllAuthenticationSets (Opnum 20) method](#section_7d8f82f7d9284ebc8afc576751e2e8d2) 149

[RRPC\_FWDeleteAllConnectionSecurityRules (Opnum 15) method](#section_ee4f85d9c07449658508c8ec96ae3b77) 144

[RRPC\_FWDeleteAllCryptoSets (Opnum 25) method](#section_67cf747ee0054630932928887df1e09d) 155

[RRPC\_FWDeleteAllFirewallRules (Opnum 8) method](#section_ca2da2f2b8ef4981abbafed5c713990d) 136

[RRPC\_FWDeleteAllMainModeRules (Opnum 35) method](#section_d47dfd57c6b9448eb5a0127d3f87cd38) 164

[RRPC\_FWDeleteAuthenticationSet (Opnum 19) method](#section_b143f0f85f274595b5310197fc491d0b) 148

[RRPC\_FWDeleteConnectionSecurityRule (Opnum 14) method](#section_b470f66f926247479be719f8aa92a611) 143

[RRPC\_FWDeleteCryptoSet (Opnum 24) method](#section_6c5b4c64545c4b108208414de327ead9) 154

[RRPC\_FWDeleteFirewallRule (Opnum 7) method](#section_e1ad0b524218420bb982d5035f6c8da3) 136

[RRPC\_FWDeleteMainModeRule (Opnum 34) method](#section_7523b410086443278bb002d00ffa6d2b) 163

[RRPC\_FWDeletePhase1SAs (Opnum 29) method](#section_0666c872b08f4b82a263e1935b763141) 159

[RRPC\_FWDeletePhase2SAs (Opnum 30) method](#section_1b6a54e35aba4bc49b9629b5514d585f) 160

[RRPC\_FWEnumAdapters (Opnum 43) method](#section_c5c35afeb29e433bb7bcca0ffa9adcd4) 171

[RRPC\_FWEnumAuthenticationSets (Opnum 21) method](#section_2dc9c31538e842d69c88927dea828314) 150

[RRPC\_FWEnumAuthenticationSets2\_10 (Opnum 54) method](#section_a788ee5f53964fd19d807e31d639b770) 184

[RRPC\_FWEnumAuthenticationSets2\_20 (Opnum 64) method](#section_be14dfc72ce64197af85575d8e96873d) 194

[RRPC\_FWEnumConnectionSecurityRules (Opnum 16) method](#section_f0f85db355104b73a4cb76bd440eaa06) 145

[RRPC\_FWEnumConnectionSecurityRules2\_10 (Opnum 51) method](#section_8e2bce0a21e4412897473b249c0b0091) 181

[RRPC\_FWEnumConnectionSecurityRules2\_20 (Opnum 60) method](#section_be8c8ed3570e45c0b063bffeec096ede) 190

[RRPC\_FWEnumCryptoSets (Opnum 26) method](#section_63e911eafbca409689a95d84c37daa56) 156

[RRPC\_FWEnumCryptoSets2\_10 (Opnum 57) method](#section_d036149f27ed45d1a55df7881b50b3f1) 187

[RRPC\_FWEnumFirewallRules (Opnum 9) method](#section_469f8466bb5c4af097a8335f648c70d7) 137

[RRPC\_FWEnumFirewallRules2\_10 (Opnum 48) method](#section_77e5217a46d8440ab330679a0939c9ae) 177

[RRPC\_FWEnumFirewallRules2\_20 (Opnum 68) method](#section_647f1dac0be1415896c906b1e03c6bdb) 198

[RRPC\_FWEnumFirewallRules2\_24 (Opnum 72) method](#section_ee81af2bc01c4cf3812ebab6c89bd6ed) 203

[RRPC\_FWEnumFirewallRules2\_25 (Opnum 76) method](#section_17e6234cbd8e48439a2f884801ceddc9) 207

[RRPC\_FWEnumFirewallRules2\_26 (Opnum 80) method](#section_cd34e45d4a254c928d34a07f3082c28e) 211

[RRPC\_FWEnumFirewallRules2\_27 (Opnum 84) method](#section_539b1f29aba74d75b034b5f9db68bf81) 215

[RRPC\_FWEnumMainModeRules (Opnum 36) method](#section_70ad911aa488474289549b9d60af3075) 165

[RRPC\_FWEnumNetworks (Opnum 42) method](#section_b3f615b50a51424cab3c15b3fb350ee1) 171

[RRPC\_FWEnumPhase1SAs (Opnum 27) method](#section_c97e073507544e688c82fb3b4275d085) 157

[RRPC\_FWEnumPhase2SAs (Opnum 28) method](#section_2c4f14fc0fee41f0b1f3e0082c929995) 158

[RRPC\_FWEnumProducts (Opnum 31) method](#section_b7a0c4f8f7a948f1b542a2937e269ca3) 161

[RRPC\_FWGetConfig (Opnum 10) method](#section_debe35e3cba149ceb331374d4e88d0ae) 138

[RRPC\_FWGetConfig2\_10 (Opnum 45) method](#section_3de45ae3d1c741a88061c435cf72e480) 174

[RRPC\_FWGetGlobalConfig (Opnum 3) method](#section_10c07a562a4f4f068510c6f3bfe5ca4a) 131

[RRPC\_FWGetGlobalConfig2\_10 (Opnum 44) method](#section_e479f36dc2254c1cbb67d46c43dc7c7e) 172

[RRPC\_FWOpenPolicyStore (Opnum 0) method](#section_2157e39a1bf04e0cabae0811fd918b11) 129

[RRPC\_FWQueryAuthenticationSets (Opnum 40) method](#section_1b4f80300bb24a1ca3e57151d5f9616b) 169

[RRPC\_FWQueryAuthenticationSets2\_20 (Opnum 65) method](#section_71836286fe1346a6a81ffed593a0fb77) 195

[RRPC\_FWQueryConnectionSecurityRules (Opnum 38) method](#section_b092c4bcf6d54f44b792599839c9db2f) 167

[RRPC\_FWQueryConnectionSecurityRules2\_20 (Opnum 61) method](#section_ce8fd2ce757a4ffa869d7ba9056bf68d) 191

[RRPC\_FWQueryCryptoSets (Opnum 41) method](#section_3dc9e91dd0ee4e1ca30d70c89052de84) 170

[RRPC\_FWQueryFirewallRules (Opnum 37) method](#section_3422b6ac5b04456dbc4ffd97e1384c3a) 166

[RRPC\_FWQueryFirewallRules2\_20 (Opnum 69) method](#section_7d1c340631a3440fa6d47232dd8bb36e) 199

[RRPC\_FWQueryFirewallRules2\_24 (Opnum 73) method](#section_0c82d80e54e34daf9cff4a92a54eaa84) 204

[RRPC\_FWQueryFirewallRules2\_25 (Opnum 77) method](#section_d6e826512e294f9999f13e2bd7cb93bd) 208

[RRPC\_FWQueryFirewallRules2\_26 (Opnum 81) method](#section_cf0aae4fbb2a42929bffd10b0b89f36e) 212

[RRPC\_FWQueryFirewallRules2\_27 (Opnum 85) method](#section_5d9f60b806f94a0585f9a54383265679) 216

[RRPC\_FWQueryMainModeRules (Opnum 39) method](#section_dc7a4aa81cb34c1f8506e6ca581d4434) 168

[RRPC\_FWRestoreDefaults (Opnum 2) method](#section_764123cab72743229fe1de1fd859fac1) 130

[RRPC\_FWSetAuthenticationSet (Opnum 18) method](#section_5a5b0f3ad8ac45608a6a485ece2cbb6c) 147

[RRPC\_FWSetAuthenticationSet2\_10 (Opnum 53) method](#section_dfb2e973f6c64b57908afa094d90ff83) 183

[RRPC\_FWSetAuthenticationSet2\_20 (Opnum 63) method](#section_4023224c13d14df4995761b0a62937ed) 193

[RRPC\_FWSetConfig (Opnum 11) method](#section_8621c5249ccc495c9c80a0830282ccf8) 140

[RRPC\_FWSetConnectionSecurityRule (Opnum 13) method](#section_c66ad9a72ee44d82a62e73ce5599c05c) 142

[RRPC\_FWSetConnectionSecurityRule2\_10 (Opnum 50) method](#section_4229bd7163c946a1a97e3c061604d717) 180

[RRPC\_FWSetConnectionSecurityRule2\_20 (Opnum 59) method](#section_b255739072f24d1da7db790853f2df36) 189

[RRPC\_FWSetCryptoSet (Opnum 23) method](#section_423d057609384c818c9beacafc69504c) 152

[RRPC\_FWSetCryptoSet2\_10 (Opnum 56) method](#section_0b68e0c61c2844eb93a06406746d8814) 186

[RRPC\_FWSetFirewallRule (Opnum 6) method](#section_676f90d026b44ed1a5eb1f88db8676b4) 135

[RRPC\_FWSetFirewallRule2\_10 (Opnum 47) method](#section_27a5eb1dc5b94ccaa50263e75df80917) 176

[RRPC\_FWSetFirewallRule2\_20 (Opnum 67) method](#section_cb987e3d916f4902aaba4b99df2e3af3) 197

[RRPC\_FWSetFirewallRule2\_24 (Opnum 71) method](#section_f6e7c7a0ffc74f72a20a0b57e784bd09) 202

[RRPC\_FWSetFirewallRule2\_25 (Opnum 75) method](#section_7d89953750424032bbdc50cec6656c2c) 206

[RRPC\_FWSetFirewallRule2\_26 (Opnum 79) method](#section_d8b1914636d14a479d31bf0194f2b9da) 210

[RRPC\_FWSetFirewallRule2\_27 (Opnum 83) method](#section_0ab23c3cdfd1452d9026115ac71dc9ae) 214

[RRPC\_FWSetGlobalConfig (Opnum 4) method](#section_45676b5c298b413b8fac0770c3def938) 132

[RRPC\_FWSetMainModeRule (Opnum 33) method](#section_268116a5af2b407b850c1380bece5169) 162

[sequencing rules](#section_b0c9335283714d63abcab0cc8dbcc3d7) 121

[timer events](#section_d55935b3e16c4bbb8ea6a89559a14509) 217

[timers](#section_16205cd8bf7b43b3aae491e558720b60) 120

[Standards assignments](#section_2251f8b0a8364098817dd5b0d025bf5e) 17

T

Timer events

[client](#section_ffc97905f3ca4bf5aa1cbed456765054) 220

[server](#section_d55935b3e16c4bbb8ea6a89559a14509) 217

Timers

[client](#section_38e9cda4e67b4be483d61c3dc7ca7b4b) 220

[server](#section_16205cd8bf7b43b3aae491e558720b60) 120

[Tracking changes](#section_acbbce890ed343c386f0f26b9285863c) 296

[Transport](#section_81eb95d6df6349b6905265c99664e71f) 18

V

[Vendor-extensible fields](#section_d8ff8b08554f4c13805f785583a16e78) 17

[Versioning](#section_6b14b8f991ab4f09873f9e2334196cdc) 16