**[MS-UNMP]:**

**User Name Mapping 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.
* **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.

**Revision Summary**

| Date | Revision History | Revision Class | Comments |
| --- | --- | --- | --- |
| 3/2/2007 | 1.0 | New | Version 1.0 release |
| 4/3/2007 | 1.1 | Minor | Version 1.1 release |
| 5/11/2007 | 1.2 | Minor | Version 1.2 release |
| 7/3/2007 | 2.0 | Major | Changed to unified format; updated technical content. |
| 8/10/2007 | 2.0.1 | Editorial | Changed language and formatting in the technical content. |
| 9/28/2007 | 3.0 | Major | Added and deleted sections; revised technical content. |
| 10/23/2007 | 3.0.1 | Editorial | Changed language and formatting in the technical content. |
| 1/25/2008 | 3.0.2 | Editorial | Changed language and formatting in the technical content. |
| 3/14/2008 | 3.0.3 | Editorial | Changed language and formatting in the technical content. |
| 6/20/2008 | 3.1 | Minor | Clarified the meaning of the technical content. |
| 7/25/2008 | 3.1.1 | Editorial | Changed language and formatting in the technical content. |
| 8/29/2008 | 4.0 | Major | Updated and revised the technical content. |
| 10/24/2008 | 5.0 | Major | Updated and revised the technical content. |
| 12/5/2008 | 6.0 | Major | Updated and revised the technical content. |
| 1/16/2009 | 6.0.1 | Editorial | Changed language and formatting in the technical content. |
| 2/27/2009 | 6.0.2 | Editorial | Changed language and formatting in the technical content. |
| 4/10/2009 | 6.0.3 | Editorial | Changed language and formatting in the technical content. |
| 5/22/2009 | 6.0.4 | Editorial | Changed language and formatting in the technical content. |
| 7/2/2009 | 6.0.5 | Editorial | Changed language and formatting in the technical content. |
| 8/14/2009 | 6.0.6 | Editorial | Changed language and formatting in the technical content. |
| 9/25/2009 | 6.1 | Minor | Clarified the meaning of the technical content. |
| 11/6/2009 | 6.1.1 | Editorial | Changed language and formatting in the technical content. |
| 12/18/2009 | 6.1.2 | Editorial | Changed language and formatting in the technical content. |
| 1/29/2010 | 6.1.3 | Editorial | Changed language and formatting in the technical content. |
| 3/12/2010 | 6.1.4 | Editorial | Changed language and formatting in the technical content. |
| 4/23/2010 | 6.1.5 | Editorial | Changed language and formatting in the technical content. |
| 6/4/2010 | 6.1.6 | Editorial | Changed language and formatting in the technical content. |
| 7/16/2010 | 6.1.6 | None | No changes to the meaning, language, or formatting of the technical content. |
| 8/27/2010 | 6.1.6 | None | No changes to the meaning, language, or formatting of the technical content. |
| 10/8/2010 | 6.1.6 | None | No changes to the meaning, language, or formatting of the technical content. |
| 11/19/2010 | 6.1.6 | None | No changes to the meaning, language, or formatting of the technical content. |
| 1/7/2011 | 6.2 | Minor | Clarified the meaning of the technical content. |
| 2/11/2011 | 6.2 | None | No changes to the meaning, language, or formatting of the technical content. |
| 3/25/2011 | 7.0 | Major | Updated and revised the technical content. |
| 5/6/2011 | 7.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 6/17/2011 | 7.1 | Minor | Clarified the meaning of the technical content. |
| 9/23/2011 | 7.1 | None | No changes to the meaning, language, or formatting of the technical content. |
| 12/16/2011 | 8.0 | Major | Updated and revised the technical content. |
| 3/30/2012 | 8.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 7/12/2012 | 9.0 | Major | Updated and revised the technical content. |
| 10/25/2012 | 9.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 1/31/2013 | 9.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 8/8/2013 | 10.0 | Major | Updated and revised the technical content. |
| 11/14/2013 | 10.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 2/13/2014 | 10.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 5/15/2014 | 10.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 6/30/2015 | 11.0 | Major | Significantly changed the technical content. |
| 10/16/2015 | 11.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 7/14/2016 | 11.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 6/1/2017 | 11.0 | None | No changes to the meaning, language, or formatting of the technical content. |
| 9/15/2017 | 12.0 | Major | Significantly changed the technical content. |

Table of Contents

[1 Introduction 7](#_Toc492420264)

[1.1 Glossary 7](#_Toc492420265)

[1.2 References 9](#_Toc492420266)

[1.2.1 Normative References 9](#_Toc492420267)

[1.2.2 Informative References 9](#_Toc492420268)

[1.3 Overview 9](#_Toc492420269)

[1.4 Relationship to Other Protocols 10](#_Toc492420270)

[1.5 Prerequisites/Preconditions 11](#_Toc492420271)

[1.6 Applicability Statement 11](#_Toc492420272)

[1.7 Versioning and Capability Negotiation 11](#_Toc492420273)

[1.8 Vendor-Extensible Fields 11](#_Toc492420274)

[1.9 Standards Assignments 11](#_Toc492420275)

[2 Messages 12](#_Toc492420276)

[2.1 Transport 12](#_Toc492420277)

[2.2 Message Syntax 12](#_Toc492420278)

[2.2.1 User Name Mapping Protocol Message Headers 12](#_Toc492420279)

[2.2.1.1 SUNRPC Request Header 12](#_Toc492420280)

[2.2.1.2 SUNRPC Response Header 12](#_Toc492420281)

[2.2.2 Common User Name Mapping Protocol Data Types 12](#_Toc492420282)

[2.2.2.1 Sizes 12](#_Toc492420283)

[2.2.2.2 MapSvrMBCSNameString 13](#_Toc492420284)

[2.2.2.3 MapSvrUnicodeNameString 13](#_Toc492420285)

[2.2.2.4 MapSvrMBCSWindowsNameString 13](#_Toc492420286)

[2.2.2.5 MapSvrUnicodeWindowsNameString 13](#_Toc492420287)

[2.2.2.6 MapSvrMBCSMapString 13](#_Toc492420288)

[2.2.2.7 MapSvrUnicodeMapString 15](#_Toc492420289)

[2.2.2.8 unix\_account 15](#_Toc492420290)

[2.2.2.9 unix\_accountW 16](#_Toc492420291)

[2.2.2.10 unix\_user\_auth 16](#_Toc492420292)

[2.2.2.11 unix\_user\_authW 17](#_Toc492420293)

[2.2.2.12 windows\_creds 17](#_Toc492420294)

[2.2.2.13 windows\_credsW 18](#_Toc492420295)

[2.2.2.14 windows\_account 18](#_Toc492420296)

[2.2.2.15 windows\_accountW 18](#_Toc492420297)

[2.2.2.16 unix\_auth 19](#_Toc492420298)

[2.2.2.17 unix\_authW 19](#_Toc492420299)

[2.2.2.18 unix\_creds 19](#_Toc492420300)

[2.2.2.19 unix\_credsW 20](#_Toc492420301)

[2.2.2.20 dump\_map\_req 20](#_Toc492420302)

[2.2.2.21 sequence\_number 20](#_Toc492420303)

[2.2.2.22 mapping\_record 21](#_Toc492420304)

[2.2.2.23 sid 21](#_Toc492420305)

[2.2.2.24 mapping\_recordW 22](#_Toc492420306)

[2.2.3 Non-XDR-Compliant Data Structures 22](#_Toc492420307)

[2.2.3.1 mapping 22](#_Toc492420308)

[2.2.3.2 maps 23](#_Toc492420309)

[2.2.3.3 mappingW 23](#_Toc492420310)

[2.2.3.4 mapsW 23](#_Toc492420311)

[2.2.4 Standard Failure Responses 24](#_Toc492420312)

[2.2.5 User Name Mapping Protocol Messages 25](#_Toc492420313)

[2.2.5.1 MAPPROC\_NULL (PROC 0) 25](#_Toc492420314)

[2.2.5.2 GETWINDOWSCREDSFROMUNIXUSERNAME\_PROC (PROC 1) 26](#_Toc492420315)

[2.2.5.3 GETUNIXCREDSFROMNTUSERNAME\_PROC (PROC 2) 26](#_Toc492420316)

[2.2.5.4 AUTHUSINGUNIXCREDS\_PROC (PROC 3) 26](#_Toc492420317)

[2.2.5.5 DUMPALLMAPS\_PROC (PROC 4) 27](#_Toc492420318)

[2.2.5.6 GETCURRENTVERSIONTOKEN\_PROC (PROC 5) 27](#_Toc492420319)

[2.2.5.7 DUMPALLMAPSEX\_PROC (PROC 6) 28](#_Toc492420320)

[2.2.5.8 GETWINDOWSGROUPFROMUNIXGROUPNAME\_PROC (PROC 7) 28](#_Toc492420321)

[2.2.5.9 GETUNIXCREDSFROMNTGROUPNAME\_PROC (PROC 8) 28](#_Toc492420322)

[2.2.5.10 GETUNIXCREDSFROMNTUSERSID\_PROC (PROC 9) 29](#_Toc492420323)

[2.2.5.11 DUMPALLMAPSW\_PROC (PROC 10) 29](#_Toc492420324)

[2.2.5.12 DUMPALLMAPSEXW\_PROC (PROC 11) 29](#_Toc492420325)

[2.2.5.13 GETWINDOWSUSERFROMUNIXUSERNAMEW\_PROC (PROC 12) 30](#_Toc492420326)

[2.2.5.14 GETUNIXCREDSFROMNTUSERNAMEW\_PROC (PROC 13) 30](#_Toc492420327)

[2.2.5.15 AUTHUSINGUNIXCREDSW\_PROC (PROC 14) 31](#_Toc492420328)

[2.2.5.16 GETWINDOWSGROUPFROMUNIXGROUPNAMEW\_PROC (PROC 15) 31](#_Toc492420329)

[2.2.5.17 GETUNIXCREDSFROMNTGROUPNAMEW\_PROC (PROC 16) 31](#_Toc492420330)

[2.2.5.18 GETUNIXCREDSFROMNTUSERSIDW\_PROC (PROC 17) 32](#_Toc492420331)

[3 Protocol Details 33](#_Toc492420332)

[3.1 Client Details 33](#_Toc492420333)

[3.1.1 Abstract Data Model 33](#_Toc492420334)

[3.1.2 Timers 34](#_Toc492420335)

[3.1.3 Initialization 34](#_Toc492420336)

[3.1.4 Higher-Layer Triggered Events 34](#_Toc492420337)

[3.1.5 Message Processing Events and Sequencing Rules 34](#_Toc492420338)

[3.1.5.1 Making the Initial Account Mapping Request to the Server 35](#_Toc492420339)

[3.1.5.2 Processing the Account Mapping Response from the Server 35](#_Toc492420340)

[3.1.5.3 Making Further Account Mapping Requests to the Server 35](#_Toc492420341)

[3.1.5.4 Polling for Cache Consistency 35](#_Toc492420342)

[3.1.6 Timer Events 36](#_Toc492420343)

[3.1.7 Local Events 36](#_Toc492420344)

[3.2 Server Details 36](#_Toc492420345)

[3.2.1 Abstract Data Model 36](#_Toc492420346)

[3.2.2 Timers 37](#_Toc492420347)

[3.2.3 Initialization 37](#_Toc492420348)

[3.2.4 Higher-Layer Triggered Events 37](#_Toc492420349)

[3.2.5 Message Processing Events and Sequencing Rules 37](#_Toc492420350)

[3.2.5.1 Processing for All Procedures 37](#_Toc492420351)

[3.2.5.2 Processing of DUMPALLMAPSXXX\_PROC Request and GETCURRENTVERSIONTOKEN\_PROC Request 37](#_Toc492420352)

[3.2.5.2.1 Processing the Initial Account Mapping Request from the Client 37](#_Toc492420353)

[3.2.5.2.2 Processing Further Account Mapping Requests from the Client 37](#_Toc492420354)

[3.2.5.2.3 Processing the Client Account Mapping Cache Refresh 38](#_Toc492420355)

[3.2.6 Timer Events 38](#_Toc492420356)

[3.2.7 Other Local Events 38](#_Toc492420357)

[4 Protocol Examples 39](#_Toc492420358)

[4.1 GETWINDOWSCREDSFROMUNIXUSERNAME\_PROC 39](#_Toc492420359)

[4.2 GETUNIXCREDSFROMNTUSERNAME\_PROC 40](#_Toc492420360)

[4.3 AUTHUSINGUNIXCREDS\_PROC 41](#_Toc492420361)

[4.4 DUMPALLMAPS\_PROC 42](#_Toc492420362)

[4.5 GETCURRENTVERSIONTOKEN\_PROC 45](#_Toc492420363)

[4.6 DUMPALLMAPSEX\_PROC 45](#_Toc492420364)

[4.7 GETWINDOWSGROUPFROMUNIXGROUPNAME\_PROC 47](#_Toc492420365)

[4.8 GETUNIXCREDSFROMNTGROUPNAME\_PROC 48](#_Toc492420366)

[4.9 GETUNIXCREDSFROMNTUSERSID\_PROC 49](#_Toc492420367)

[4.10 DUMPALLMAPSW\_PROC 50](#_Toc492420368)

[4.11 DUMPALLMAPSEXW\_PROC 52](#_Toc492420369)

[4.12 GETWINDOWSUSERFROMUNIXUSERNAMEW\_PROC 53](#_Toc492420370)

[4.13 GETUNIXCREDSFROMNTUSERNAMEW\_PROC 54](#_Toc492420371)

[4.14 AUTHUSINGUNIXCREDSW\_PROC 55](#_Toc492420372)

[4.15 GETWINDOWSGROUPFROMUNIXGROUPNAMEW\_PROC 56](#_Toc492420373)

[4.16 GETUNIXCREDSFROMNTGROUPNAMEW\_PROC 57](#_Toc492420374)

[4.17 GETUNIXCREDSFROMNTUSERSIDW\_PROC 58](#_Toc492420375)

[5 Security 60](#_Toc492420376)

[5.1 Security Considerations for Implementers 60](#_Toc492420377)

[5.2 Index of Security Parameters 60](#_Toc492420378)

[6 Appendix A: Full SunRPC IDL 61](#_Toc492420379)

[7 Appendix B: Sample Code to Encode and Decode Non-XDR-Compliant Data Types 64](#_Toc492420380)

[7.1 Header File Content 64](#_Toc492420381)

[7.2 Encode/Decode Routines For Non-XDR Data Types Using XDR Primitives 65](#_Toc492420382)

[8 Appendix C: Product Behavior 67](#_Toc492420383)

[9 Change Tracking 69](#_Toc492420384)

[10 Index 70](#_Toc492420385)

# Introduction

The Windows and [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) operating systems use different mechanisms for user identification, authentication, and resource access control. Users have separate accounts in the Windows portion and the UNIX portion of any network. Because Windows and UNIX user identifications and user names are stored and used differently, there is no association between the two sets, even though the same users exist on each network.

The User Name Mapping Protocol maps Windows domain user and group account names (DOMAIN\NAME) to the [**POSIX**](#gt_b1c15a13-c6a7-40bb-a0db-1947a755118d) user and [**group identifiers**](#gt_323122fc-2cf0-42be-98c6-cb492663c44b) (UIDs and GIDs) utilized in AUTH\_UNIX authentication and vice versa. This enables the association of user names for users who have different identities in Windows-based and UNIX-based [**domains**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca). For example, this protocol allows user and group accounts from multiple Windows domains to access resources on [**Network File System (NFS)**](#gt_f68d64c0-6af5-4de4-8e2e-18b548aea7da) file servers by using UIDs and GIDs. The User Name Mapping Protocol supports only retrieval of mappings; it does not include procedures for changing user mappings.

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:

**advanced map**: Used to map accounts that have different names on the [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) and Windows systems. [**Advanced maps**](#gt_2ee13d81-4d2a-44a0-8df0-aa38bc9f556d) are also used to map users from different Windows domains, and they can also explicitly map accounts that would generally be mapped by [**simple maps**](#gt_f00df346-27f8-48be-bd68-f5d72c13a565). For more information, see [[NFSAUTH]](https://go.microsoft.com/fwlink/?LinkId=90231).

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

**DUMPALLMAPSXXX\_PROC**: A reference to the following procedures: DUMPALLMAPS\_PROC, DUMPALLMAPSEX\_PROC, DUMPALLMAPSW\_PROC, and DUMPALLMAPSEXW\_PROC.

**group identifier (group ID or GID)**: A number that identifies a group of users to a [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) operating system. The scope of the number is at least machine-wide but can also be coordinated across a group of machines by means of services, such as the Network Information Service (NIS).

**group map**: An association between a Windows group account name, a [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) group account name, and a [**GID**](#gt_323122fc-2cf0-42be-98c6-cb492663c44b).

**little-endian**: Multiple-byte values that are byte-ordered with the least significant byte stored in the memory location with the lowest address.

**map**: An association between a Windows-based network user or group name and a [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed)-based network user or group name.

**multibyte character set (MBCS)**: An alternative to [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) for supporting character sets, like Japanese and Chinese, that cannot be represented in a single byte. Under MBCS, characters are encoded in either one or two bytes. In two-byte characters, the first byte, or "lead" byte, signals that both it and the following byte are to be interpreted as one character. The first byte comes from a range of codes reserved for use as lead bytes. Which ranges of bytes can be lead bytes depends on the code page in use. For example, Japanese code page 932 uses the range 0x81 through 0x9F as lead bytes, but Korean code page 949 uses a different range.

**Network File System (NFS)**: A [**Network File System**](#gt_f68d64c0-6af5-4de4-8e2e-18b548aea7da) protocol, as specified in [[RFC1094]](https://go.microsoft.com/fwlink/?LinkId=90267) and [[RFC1813]](https://go.microsoft.com/fwlink/?LinkId=90294). This protocol is compatible with [**NFS**](#gt_f68d64c0-6af5-4de4-8e2e-18b548aea7da) version 3 (NFSv3). [**NFS**](#gt_f68d64c0-6af5-4de4-8e2e-18b548aea7da) version 4 (NFSv4) obviates the need for this protocol by allowing Windows and [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) [**domains**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca) to interoperate using Kerberos version 5, which allows them to share the same namespace.

**OEMCP**: The default OEM code page of the system. The OEM code page is used for conversions of MS-DOS–based, text-mode applications.

**portmapper service**: A [**portmapper service**](#gt_2335ad8d-af11-486e-8f9b-be681379610d) is a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) service that provides discovery services; clients of the [**portmapper service**](#gt_2335ad8d-af11-486e-8f9b-be681379610d) can use it to discover other [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) services running on the same computer. The information returned by the [**portmapper service**](#gt_2335ad8d-af11-486e-8f9b-be681379610d) is then used by the client of the [**portmapper service**](#gt_2335ad8d-af11-486e-8f9b-be681379610d) to act as a client for the discovered [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) service. The [**portmapper service**](#gt_2335ad8d-af11-486e-8f9b-be681379610d) runs on a specific well-known port (Port 111 on TCP/UDP).

**POSIX**: Portable Operating System Interface, as specified in [[IEEE1003.1]](https://go.microsoft.com/fwlink/?LinkId=89897). [**POSIX**](#gt_b1c15a13-c6a7-40bb-a0db-1947a755118d) is a set of standard operating system interfaces based on [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed). This term is used interchangeably with [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) in the rest of this document, as described in [IEEE1003.1].

**primary map**: When multiple Windows accounts are mapped to a single [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account, one of these [**mappings**](#gt_173464f7-c0e8-4181-941a-f7df6725be5a) can be designated as a "primary" [**mapping**](#gt_173464f7-c0e8-4181-941a-f7df6725be5a). For more information, see [NFSAUTH].

**security identifier (SID)**: An identifier for security principals that is used to identify an account or a group. Conceptually, the [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) is composed of an account authority portion (typically a [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca)) and a smaller integer representing an identity relative to the account authority, termed the relative identifier (RID). The [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) format is specified in [[MS-DTYP]](%5BMS-DTYP%5D.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.2; a string representation of [**SIDs**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) is specified in [MS-DTYP] section 2.4.2 and [[MS-AZOD]](%5BMS-AZOD%5D.pdf#Section_5a0a0a3ec7a742e1b5f2cc8d8bd9739e) section 1.1.1.2.

**simple map**: [**Maps**](#gt_173464f7-c0e8-4181-941a-f7df6725be5a) between accounts with the exact same name in [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) as in Windows. For more information, see [NFSAUTH].

**SUNRPC**: A remote procedure call (RPC) protocol from Sun Microsystems [[RFC1057]](https://go.microsoft.com/fwlink/?LinkId=90265). [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) forms the basis of the [**Network File System (NFS)**](#gt_f68d64c0-6af5-4de4-8e2e-18b548aea7da) Protocol. [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) has no relationship to Remote Procedure Call Protocol Extensions, as specified in [[MS-RPCE]](%5BMS-RPCE%5D.pdf#Section_290c38b192fe422991e64fc376610c15).

**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).

**UNIX**: A multiuser, multitasking operating system developed at Bell Laboratories in the 1970s. In this document, the term "[**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed)" is used to refer to any derivatives of this operating system.

**user identifier (user ID or UID)**: A number that identifies a particular user to a [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) operating system. The scope of the number is at least machine-wide and can be coordinated across a group of machines by means of services such as NIS.

**user map**: An association between a Windows user account name, a [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) user account name, and a UID.

**wide characters**: Characters represented by a 2-byte value that are encoded using [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) UTF-16. Unless otherwise stated, no range restrictions apply.

**XDR**: The data encoding standard used by [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) for a selection of common data types such as strings, integers, and arrays of integers, as specified in [[RFC4506]](https://go.microsoft.com/fwlink/?LinkId=90478).

**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. We will assist you in finding the relevant information.

[IEEE1003.1] The Open Group, "IEEE Std 1003.1, 2004 Edition", 2004, [http://www.unix.org/version3/ieee\_std.html](https://go.microsoft.com/fwlink/?LinkId=89897)

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

[RFC1057] Sun Microsystems, Inc., "RPC: Remote Procedure Call Protocol Specification Version 2", RFC 1057, June 1998, [http://www.ietf.org/rfc/rfc1057.txt](https://go.microsoft.com/fwlink/?LinkId=90265)

[RFC1831] Srinivasan, R., "RPC: Remote Procedure Call Protocol Specification Version 2", RFC 1831, August 1995, [http://www.ietf.org/rfc/rfc1831.txt](https://go.microsoft.com/fwlink/?LinkId=90295)

[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)

[RFC4506] Network Appliance, Inc., "XDR: External Data Representation Standard", STD 67, RFC 4506, May 2006, [http://www.ietf.org/rfc/rfc4506.txt](https://go.microsoft.com/fwlink/?LinkId=90478)

### Informative References

[NFSAUTH] Russel, C., "NFS Authentication", [http://www.microsoft.com/technet/interopmigration/unix/sfu/nfsauth.mspx](https://go.microsoft.com/fwlink/?LinkId=90231)

[NIS] Sun Microsystems, Inc., "System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)", [http://docs.sun.com/app/docs/doc/816-4556](https://go.microsoft.com/fwlink/?LinkId=90234)

[WINNSP] Microsoft Corporation, "Namespace Planning for DNS", January 2005, [http://technet2.microsoft.com/WindowsServer/en/library/8ec96981-6b1a-48ec-bd3e-d8d43bc814311033.mspx](https://go.microsoft.com/fwlink/?LinkId=90567)

[WINUGA] Microsoft Corporation, "Creating User and Group Accounts", [http://www.microsoft.com/technet/archive/winntas/deploy/confeat/05wntpca.mspx](https://go.microsoft.com/fwlink/?LinkId=90568)

## Overview

The User Name Mapping Protocol maps Windows domain identities (user and group account names) to [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) user and UNIX group identities (user and group account names and their corresponding UID and [**GID**](#gt_323122fc-2cf0-42be-98c6-cb492663c44b)) and vice versa. Clients of the User Name Mapping Protocol use [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600)-formatted messages to enumerate and/or translate user and group account information between a UNIX and a Windows domain. The User Name Mapping Protocol exists to allow a one-to-one mapping of each Windows group account name to a GID number and a one-to-one mapping of each Windows user account name to a [**user identifier (user ID or UID)**](#gt_9c7c7fef-9933-4226-87a4-ea2a48f412c2) number.

The User Name Mapping Protocol is invoked by a client application when the application needs to provide a [**user map**](#gt_6f6cefa4-0c32-4524-a267-2ce2cb29d7e9) or a [**group map**](#gt_5501e0d8-1bdc-464c-8c40-c78de7b96734) between a UNIX user or group and the corresponding Windows user or group. This need is application specific and is not specified by the User Name Mapping Protocol. The UNIX or Windows user, or UNIX or Windows group, that needs to be mapped is supplied to the User Name Mapping Protocol by the client application, and the mapped user/group is returned to the client by the User Name Mapping Protocol server. For user mapping and group mapping enumerations, the client application specifies the enumeration parameters, and the User Name Mapping Protocol server returns the enumerated user mappings/group mappings to the client.

These mapping enumerations can be cached by the client application and kept up to date by periodically polling the server to determine if the cached mappings are still valid. The User Name Mapping Protocol does not provide authentication or authorization of the application-provided user/group; to the client, it is a read-only account mapping service.

An example of this authentication behavior is a user on a UNIX machine making a file access request that contains AUTH\_UNIX–formatted user credentials to an [**NFS**](#gt_f68d64c0-6af5-4de4-8e2e-18b548aea7da) server implemented on a computer running Windows. The NFS server acts as a User Name Mapping Protocol client (or "user map") to request the Windows domain user and group names (from the User Name Mapping Protocol server) that match the AUTH\_UNIX credentials, [[RFC1057]](https://go.microsoft.com/fwlink/?LinkId=90265) section 9.2, supplied by the UNIX user. This action enables the NFS server to authenticate the file access request.

This document specifies the SUNRPC-formatted messages that provide support for the following operations:

* Mapping [**POSIX**](#gt_b1c15a13-c6a7-40bb-a0db-1947a755118d) user and group names and/or UIDs/GIDs to Windows domain and account names.
* Mapping Windows domain and account names to POSIX user and group names, and UIDs and GIDs.
* Allowing a User Name Mapping Protocol client to authenticate a POSIX user by providing a user name and password.
* Enumerating all user mappings and group mappings between POSIX accounts and Windows accounts known to the User Name Mapping Protocol server.
* Testing to see if any maps previously enumerated by a client have changed from the time of the last check.
* Mapping a Windows domain [**security identifier (SID)**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) to a POSIX user/group name and UID/GID.

This document specifies versions 1 and 2 of the User Name Mapping Protocol. Version 1 is comprised of a set of nine SUNRPC procedures; version 2 consists of a set of 18 SUNRPC procedures. For a list of these procedures, see the table in section [2.2.5](#Section_f29045a2a1c54fa9a3c96c2e35b2f1ae).

There are several differences between User Name Mapping Protocol version 1 and User Name Mapping Protocol version 2. Version 2 added procedures 10–17, which are the [**wide character**](#gt_30c30de3-1d00-4d0d-9109-fcc0094fc95a) ([**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8)) counterparts of procedures 1–4 and 6–9. Procedures 1–4 and 6–8 accept [**multibyte character set (MBCS)**](#gt_8249acae-6bac-450e-9f08-1cb90ce200bd) character-encoded strings as input. Version 2 includes the additional procedure 9, which takes a Windows account SID and returns an MBCS character-encoded UNIX account [**map**](#gt_173464f7-c0e8-4181-941a-f7df6725be5a) that corresponds to the Windows account represented by that SID. The wide character (Unicode) counterpart to procedure 9 is procedure 17.

## Relationship to Other Protocols

The User Name Mapping Protocol relies on [[RFC1057]](https://go.microsoft.com/fwlink/?LinkId=90265) and [[RFC4506]](https://go.microsoft.com/fwlink/?LinkId=90478) for communicating with clients by means of the [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) message version 2 and [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d) protocols as specified in those documents. The User Name Mapping Protocol uses SUNRPC authentication level AUTH\_NULL (as specified in [RFC1057]). The User Name Mapping Protocol uses SUNRPC message version 2 implemented on top of TCP and UDP. The User Name Mapping Protocol message formats are not sensitive to which underlying transports (TCP or UDP) are being used.[<1>](#Appendix_A_1" \o "Product behavior note 1)

## Prerequisites/Preconditions

It is required that the User Name Mapping Protocol server has been previously configured with all the appropriate [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) and Windows domain name and group mapping information, and that it has registered with the [**portmapper service**](#gt_2335ad8d-af11-486e-8f9b-be681379610d) (as specified in [[RFC1057]](https://go.microsoft.com/fwlink/?LinkId=90265) Appendix A) on the same computer as the User Name Mapping Protocol server. Normal TCP/IP services sufficient to provide TCP-based or UDP-based communications must be available.[<2>](#Appendix_A_2" \o "Product behavior note 2)

## Applicability Statement

The User Name Mapping Protocol is applicable in a heterogeneous network environment where users have separate accounts in [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) and Windows infrastructures. This protocol provides a means to associate user and group accounts in two networks for users or groups that have different identities in UNIX-based and Windows-based administrative [**domains**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).[<3>](#Appendix_A_3" \o "Product behavior note 3)

## Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

* **Protocol Versions:** The User Name Mapping Protocol supports versions 1 and 2. These dialects are defined in section [2.2](#Section_3677a9ee8f1c4ef6809f725a3154ce0b).
* **Capability Negotiation:** Version negotiation of the User Name Mapping Protocol is achieved using the standard method for protocol negotiation for [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) services as specified in [[RFC1057]](https://go.microsoft.com/fwlink/?LinkId=90265) section 8. The User Name Mapping Protocol client requests a specific version of the User Name Mapping Protocol from the [**portmapper service**](#gt_2335ad8d-af11-486e-8f9b-be681379610d) (as specified in [RFC1057] Appendix A). The portmapper service replies with the available versions registered by the User Name Mapping Protocol server. It is recommended that requests made to the User Name Mapping Protocol server for versions other than those supported with a SUNRPC PROG\_MISMATCH message, as specified in [RFC1057].

## Vendor-Extensible Fields

None.

## Standards Assignments

|  Parameter  |  Value  |  Reference  |
| --- | --- | --- |
| MAPSVC\_PROGRAM | 351455 | [[RFC1057]](https://go.microsoft.com/fwlink/?LinkId=90265) section 7.3 |

# Messages

## Transport

The User Name Mapping Protocol is a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) protocol (as specified in [[RFC1057]](https://go.microsoft.com/fwlink/?LinkId=90265)) that runs on TCP/IP using TCP and/or UDP transports, with a well-known program number of MAPSVC\_PROGRAM (351455). The User Name Mapping Protocol server registers an available TCP/UDP port with the local [**portmapper service**](#gt_2335ad8d-af11-486e-8f9b-be681379610d) on startup using the MAPSVC\_PROGRAM number for all combinations of TCP, UDP, and protocol versions that the User Name Mapping Protocol server is capable of or configured to accept. User Name Mapping Protocol clients query the SUNRPC portmapper server for the TCP/UDP port number on which the User Name Mapping Protocol is registered and listening for the requested version and transport combination.

Configuration of the portmapper service and port registration is specified in [RFC1057] Appendix A. The User Name Mapping Protocol does not define a configuration interface to the portmapper service.

The User Name Mapping Protocol server provides a procedure-oriented interface to the User Name Mapping Protocol clients. Clients identify the remote procedure by using a combination of a 32-bit program number, a 32-bit version number, and a 32-bit procedure number (as specified in [RFC1057]). The service is stateless; every SUNRPC call is self-contained and does not depend on the previous calls made or previous state of the service.

The User Name Mapping Protocol server accepts all SUNRPC packets with an authentication level of AUTH\_NULL, as specified in [RFC1057] section 9.1. Therefore, no authentication information is required by the client.

## Message Syntax

The following structures are specified in [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d) Data Definition Language syntax (as specified in [[RFC4506]](https://go.microsoft.com/fwlink/?LinkId=90478) section 6) while procedures are defined in the [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) language, as specified in [[RFC1057]](https://go.microsoft.com/fwlink/?LinkId=90265) section 11.

### User Name Mapping Protocol Message Headers

#### SUNRPC Request Header

The User Name Mapping Protocol uses standard [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) version 2 msg\_type CALL headers. Requests are made with an authentication level of AUTH\_NULL. This header format and its fields and values are specified in [[RFC1057]](https://go.microsoft.com/fwlink/?LinkId=90265) section 8.

#### SUNRPC Response Header

The User Name Mapping Protocol uses standard [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) version 2 msg\_type REPLY headers. This header format and its fields and values are specified in [[RFC1057]](https://go.microsoft.com/fwlink/?LinkId=90265) section 8.

### Common User Name Mapping Protocol Data Types

In this section, the [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d) Data Description Language (as specified in [[RFC4506]](https://go.microsoft.com/fwlink/?LinkId=90478)) is used to specify the XDR format parameters and results of each of the [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) service procedures that a User Name Mapping Protocol server provides.

#### Sizes

1. const MAXNAMELEN = 128;
2. const MAXNAMELENx2 = 256;
3. const MAXLINELEN = 256;
4. const MAXLINELENx2 = 512;
5. const MAXGIDS = 32;
6. const MAXSIDLEN = 72;

(MAXGIDS is the maximum count of [**GIDs**](#gt_323122fc-2cf0-42be-98c6-cb492663c44b). This includes both the primary GID and any supplementary GIDs.)

#### MapSvrMBCSNameString

1. typedef opaque MapSvrMBCSNameString<MAXNAMELEN>;

An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d) variable-length opaque data field, as specified in [[RFC4506]](https://go.microsoft.com/fwlink/?LinkId=90478) section 4.10, whose maximum length is specified in bytes. The length is equal to the number of [**MBCS**](#gt_8249acae-6bac-450e-9f08-1cb90ce200bd) bytes encoded in the system [**OEM code page (OEMCP)**](#gt_c2904abd-1168-4ec8-bd2c-a59b0351d1e3), including multibyte characters, as specified by the **length** field that precedes the byte stream. The value of the **length** field MUST NOT exceed the value MAXNAMELEN. Minimum length is 0.

#### MapSvrUnicodeNameString

1. typedef opaque MapSvrUnicodeNameString<MAXNAMELENx2>;

An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d) variable-length opaque data field, as specified in [[RFC4506]](https://go.microsoft.com/fwlink/?LinkId=90478) section 4.10, whose maximum length is specified in bytes. The maximum length is defined by the **length** field that precedes the byte stream. The value of the **length** field MUST NOT exceed the value MAXNAMELENx2. The maximum length of the character string is equal to as many 2-byte [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) (UTF-16) characters as can be stored in a MapSvrUnicodeNameString, with a maximum length equal to **length**. Minimum length is 0.

#### MapSvrMBCSWindowsNameString

1. typedef opaque MapSvrMBCSWindowsNameString<MAXLINELEN>;

An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d) variable-length opaque data field, as specified in [[RFC4506]](https://go.microsoft.com/fwlink/?LinkId=90478) section 4.10, whose maximum length is specified in bytes. The length is equal to the number of [**MBCS**](#gt_8249acae-6bac-450e-9f08-1cb90ce200bd) bytes encoded in the system [**OEMCP**](#gt_c2904abd-1168-4ec8-bd2c-a59b0351d1e3), including multibyte characters, as specified by the **length** field that precedes the byte stream. The value of the **length** field MUST NOT exceed the value MAXLINELEN. Minimum length is 0.

#### MapSvrUnicodeWindowsNameString

1. typedef opaque MapSvrUnicodeWindowsNameString<MAXLINELENx2>;

An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d) variable-length opaque data field, as specified in [[RFC4506]](https://go.microsoft.com/fwlink/?LinkId=90478) section 4.10, whose maximum length is specified in bytes. The maximum length is defined by the **length** field that precedes the byte stream. The value of the **length** field MUST NOT exceed the value MAXLINELENx2. The maximum length of the character string is equal to as many 2-byte [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) (UTF-16) characters as can be stored in a MapSvrUnicodeWindowsNameString, with a maximum length equal to **length**. Minimum length is 0.

#### MapSvrMBCSMapString

1. typedef opaque MapSvrMBCSMapString<MAXLINELEN>;

An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d) variable-length opaque data field, as specified in [[RFC4506]](https://go.microsoft.com/fwlink/?LinkId=90478) section 4.10, whose maximum length is specified in bytes. The length is equal to the number of [**MBCS**](#gt_8249acae-6bac-450e-9f08-1cb90ce200bd) bytes encoded in the system [**OEMCP**](#gt_c2904abd-1168-4ec8-bd2c-a59b0351d1e3), including multibyte characters, as specified by the **length** field that precedes the byte stream. The value of the **length** field MUST NOT exceed the value MAXLINELEN. Minimum length is 0.

This type is used to define a single account [**map**](#gt_173464f7-c0e8-4181-941a-f7df6725be5a) as a colon-delimited string of MBCS characters. This type is returned as an output from the map enumeration procedure. For more information, see section [2.2.3.2](#Section_07e03281db04473588930136b435ddb4).

The format of MapSvrMBCSMapString is a sequence of colon-delimited fields. It has one of two forms, depending on the context: [**user map**](#gt_6f6cefa4-0c32-4524-a267-2ce2cb29d7e9) or [**group map**](#gt_5501e0d8-1bdc-464c-8c40-c78de7b96734), as follows.

For user map, MapSvrMBCSMapString has the following format.

1. MapType:WindowsAccountName:AuthType:UNIXDomain:UNIXServer:
2. UNIXAccountName:UNIXPassword:ID:GIDArray

For group map, MapSvrMBCSMapString has the following format.

1. MapType:WindowsAccountName:AuthType:UNIXDomain:UNIXServer:
2. UNIXAccountName:GID

**MapType**: A single MBCS character that indicates the type of map from which the mapping was derived. It MUST be one of the following characters.

|  Value |  Meaning  |
| --- | --- |
| '\*' | The map is a [**primary map**](#gt_0b3ed846-9891-47a3-ac7d-0e3dea294d29). |
| '^' | The map is an [**advanced map**](#gt_2ee13d81-4d2a-44a0-8df0-aa38bc9f556d). |
| '-' | The map is a [**simple map**](#gt_f00df346-27f8-48be-bd68-f5d72c13a565). |

**WindowsAccountName**: A string of MBCS characters that contains the Windows account name. It MUST be in DOMAIN\NAME format.

**AuthType**: A single MBCS character that indicates which entity provided the map. **AuthType** MUST be one of the values in the following table. If the value is AUTH\_NIS, the source MUST be a NIS service on the network. If the value is AUTH\_FILE, the source SHOULD[<4>](#Appendix_A_4" \o "Product behavior note 4) be from the service-maintained database local to the User Name Mapping Protocol server.

|  Value |  Meaning  |
| --- | --- |
| '0' (AUTH\_FILE) | The map was obtained from a service-maintained database local to the User Name Mapping Protocol server. The form of the database is implementation-specific. |
| '1' (AUTH\_NIS) | The map was obtained from a NIS service on the network. NIS is specified in [[NIS]](https://go.microsoft.com/fwlink/?LinkId=90234). |

**UNIXDomain**: A string of MBCS characters that contains the string "PCNFS" if the map was obtained from a service-maintained database, or the NIS server domain to which the account belongs if the map was obtained from a NIS service. If **AuthType** is equal to AUTH\_NIS, this field MUST contain the NIS server domain the account is a member of.

**UNIXServer**: A string of MBCS characters that contains the string "PCNFS" if the map was obtained from a service-maintained database, or a string that represents the NIS server name to which the account belongs if the map was obtained from a NIS server.

**UNIXAccountName**: A string of MBCS characters that represents the [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account name.

**UNIXPassword**: A sequence of bytes that represents the password for a user record as returned by a call to the crypt() API that uses the user's cleartext password, as specified in [[IEEE1003.1]](https://go.microsoft.com/fwlink/?LinkId=89897) System Interfaces Volume (XSH). This field is empty when the password is not available or does not apply. The password record MUST NOT contain any MBCS colon characters.

**ID**: A string of MBCS characters that contains the ID for the UNIX account.

**GID**: A string of MBCS characters that contains the GID for the UNIX account.

**GIDArray**: A string of MBCS characters that contains the primary and supplementary [**GIDs**](#gt_323122fc-2cf0-42be-98c6-cb492663c44b) for the UNIX account, with each supplementary GID after the primary GID, and separated by additional colon characters.

#### MapSvrUnicodeMapString

1. typedef opaque MapSvrUnicodeMapString<MAXLINELENx2>;

An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d) variable-length opaque data field, as specified in [[RFC4506]](https://go.microsoft.com/fwlink/?LinkId=90478) section 4.10, whose maximum length is specified in bytes. The maximum length is defined by the **length** field that precedes the byte stream. The value of the **length** field MUST NOT exceed the value MAXLINELENx2. The maximum length of the character string is equal to as many 2-byte [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) (UTF-16) characters as can be stored in a MapSvrUnicodeMapString, with a maximum length equal to **length**. Minimum length is 0.

This type is used to define a single account map in colon-delimited string format when returned as an output from the map enumeration procedure. For more information, see section [2.2.3.4](#Section_003667ced8bb4e5eafb6a73e3340f67a).

The format of a MapSvrUnicodeMapString field is a sequence of colon-delimited fields as specified in section [2.2.2.6](#Section_d5e345407d2e42a3856a7ac74d3750e5), substituting Unicode characters for [**MBCS**](#gt_8249acae-6bac-450e-9f08-1cb90ce200bd) characters.

#### unix\_account

This type is used to specify a [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account name in [**MBCS**](#gt_8249acae-6bac-450e-9f08-1cb90ce200bd) format, in addition to an ID used to search for the corresponding Windows account information when mapping a UNIX account name to a Windows account name. For more information, see sections [2.2.5.2](#Section_887248fa50f84268a47d27583ff97cf8) and [2.2.5.8](#Section_ef1fd6724c4646d0b3b779c2fa8e62e9).

1. struct unix\_account {
2. long SearchOption;
3. long Reserved;
4. long ID;
5. MapSvrMBCSNameString UnixAccountName;
6. };

**SearchOption**: An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d)-encoded, 32-bit signed integer that defines the user search criteria to use for the request. **SearchOption** MUST be one of the following values.

|  Value |  Meaning  |
| --- | --- |
| 0x00000001 | If set, **UnixAccountName** is valid and MUST be used as the search criterion. |
| 0x00000002 | If set, **ID** is valid and MUST be used as the search criterion. |
| 0x00000003 | If set, **UnixAccountName** and **ID** are both valid and both MUST be used as the search criteria. |

**Reserved**: A 32-bit signed integer that MUST be sent as 0x00000000 and MUST be ignored on receipt.

**ID**: An XDR-encoded, 32-bit signed integer that contains the UNIX account ID to use as the search criterion. If **SearchOption** is not 0x00000002 or 0x00000003, this value MUST be ignored.

**UnixAccountName**: A [MapSvrMBCSNameString (section 2.2.2.2)](#Section_ff8f275537504d02be9ce4e3b1619b5f) that contains the name of the UNIX account to use as the search criterion. The length of the string MUST NOT exceed 128 bytes. POSIX user and group account name constraints are specified in [[IEEE1003.1]](https://go.microsoft.com/fwlink/?LinkId=89897). If **SearchOption** is not 0x00000001 or 0x00000003, this value MUST be ignored.

#### unix\_accountW

This type is used to specify a [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account name in [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) format, in addition to an ID used to search for the corresponding Windows account information when mapping a UNIX account name to a Windows account name. For more information, see sections [2.2.5.13](#Section_45e29142f3e247a19cad5d4758318bd4) and [2.2.5.16](#Section_7e7e7c29834f4f6ba651fa1388e7de0a).

1. struct unix\_accountW {
2. long SearchOption;
3. long Reserved;
4. long ID;
5. MapSvrUnicodeNameString UnixAccountName;
6. };

**SearchOption**: An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d)-encoded, 32-bit signed integer that defines the user search criteria to use for the request. **SearchOption** MUST be one of the following values.

|  Value |  Meaning  |
| --- | --- |
| 0x00000001 | If set, **UnixAccountName** is valid and MUST be used as the search criterion. |
| 0x00000002 | If set, **ID** is valid and MUST be used as the search criterion. |
| 0x00000003 | If set, **UnixAccountName** and **ID** are both valid and both MUST be used as the search criteria. |

**Reserved**: A 32-bit signed integer that MUST be sent as 0x00000000 and MUST be ignored on receipt.

**ID**: An XDR-encoded, 32-bit signed integer that contains the UNIX account ID to use as the search criterion. If **SearchOption** is not 0x00000002 or 0x00000003, this value MUST be ignored.

**UnixAccountName**: A [MapSvrUnicodeNameString (section 2.2.2.3)](#Section_dbd5ac38df0c4798a86ead90687b0ca2) that contains the name of the UNIX account to use as the search criterion. The length of the string MUST NOT exceed 256 bytes. POSIX user and group account name constraints are specified in [[IEEE1003.1]](https://go.microsoft.com/fwlink/?LinkId=89897). If **SearchOption** is not 0x00000001 or 0x00000003, this value MUST be ignored.

#### unix\_user\_auth

This type is used to specify a [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account name (in [**MBCS**](#gt_8249acae-6bac-450e-9f08-1cb90ce200bd) format) and a password to retrieve the set of UNIX account details that correspond to the account. For more information, see section [2.2.5.4](#Section_c7f40a27eb2249d7a1d77e428bc61eb9).

1. struct unix\_user\_auth {
2. MapSvrMBCSNameString UnixUserAccountName;
3. MapSvrMBCSNameString UnixUserAccountPassword;
4. };

**UnixUserAccountName**: A [MapSvrMBCSNameString (section 2.2.2.2)](#Section_ff8f275537504d02be9ce4e3b1619b5f) that contains the name of the UNIX user account to use as the search criterion. The length of this string MUST NOT exceed 128 bytes. POSIX user and group account name constraints are specified in [[IEEE1003.1]](https://go.microsoft.com/fwlink/?LinkId=89897).

**UnixUserAccountPassword**: An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d) variable-length opaque data field, as defined in [[RFC4506]](https://go.microsoft.com/fwlink/?LinkId=90478) section 4.10, that contains the password of the UNIX user account to use as the search criterion. The length of this field MUST NOT exceed 128 bytes. This string MUST be generated by a call to the [**POSIX**](#gt_b1c15a13-c6a7-40bb-a0db-1947a755118d) crypt() function, as described in section 3 of the System Interfaces Volume (XSH) of [IEEE1003.1].

#### unix\_user\_authW

This type is used to specify a [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account name (in [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) format) and a password to retrieve the set of UNIX account details that correspond to the account. For more information, see section [2.2.5.15](#Section_f80f8d46fca64f58b7e06e7f478eb121).

1. struct unix\_user\_authW {
2. MapSvrUnicodeNameString UnixUserAccountName;
3. MapSvrUnicodeNameString UnixUserAccountPassword;
4. };

**UnixUserAccountName**: A [MapSvrUnicodeNameString (section 2.2.2.3)](#Section_dbd5ac38df0c4798a86ead90687b0ca2) that contains the name of the UNIX user to use as the search criterion. The length of the string MUST NOT exceed 256 bytes. POSIX user and group account name constraints are specified in [[IEEE1003.1]](https://go.microsoft.com/fwlink/?LinkId=89897).

**UnixUserAccountPassword**: A MapSvrUnicodeNameString (section [2.2.2.3](#Section_dbd5ac38df0c4798a86ead90687b0ca2)) that contains the password of the UNIX user account to use as the search criterion. The length of the string MUST NOT exceed 256 bytes. This string MUST be generated by a call to the [**POSIX**](#gt_b1c15a13-c6a7-40bb-a0db-1947a755118d) crypt() function, as described in section 3 of the System Interfaces Volume (XSH) of [IEEE1003.1].

#### windows\_creds

This type represents the Windows account name (in [**MBCS**](#gt_8249acae-6bac-450e-9f08-1cb90ce200bd) format) when used as an output parameter from a search for the corresponding [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account name (in MBCS format) and/or UNIX ID. For more information, see sections [2.2.5.2](#Section_887248fa50f84268a47d27583ff97cf8) and [2.2.5.8](#Section_ef1fd6724c4646d0b3b779c2fa8e62e9))

1. struct windows\_creds {
2. long Status;
3. long Reserved;
4. MapSvrMBCSWindowsNameString WindowsAccountName;
5. };

**Status**: An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d)-encoded, Boolean return value. This MUST be either 0 or 1. A value of 0 indicates success; a value of 1 indicates failure.

**Reserved**: A 32-bit signed integer that MUST be 0x00000000 and MUST be ignored on receipt.

**WindowsAccountName**: A [MapSvrMBCSWindowsNameString (section 2.2.2.4)](#Section_86c54411d47d4ff8993d8bf7d3111717) that contains the name of the mapped Windows user or group account that MUST be in the form "DOMAIN\NAME". The length of the string MUST NOT exceed 256 bytes. Windows user/group account name constraints are specified in [[WINUGA]](https://go.microsoft.com/fwlink/?LinkId=90568), and Windows domain naming conventions are specified in [[WINNSP]](https://go.microsoft.com/fwlink/?LinkId=90567). If **Status** does not equal 0x00000000, this value MUST be ignored.

#### windows\_credsW

This type represents the Windows account name (in [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) format) when used as an output parameter from a search for the corresponding [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account name (in Unicode format) and/or UNIX ID. For more information, see sections [2.2.5.13](#Section_45e29142f3e247a19cad5d4758318bd4) and [2.2.5.16](#Section_7e7e7c29834f4f6ba651fa1388e7de0a).

1. struct windows\_credsW {
2. long Status;
3. long Reserved;
4. MapSvrUnicodeWindowsNameString WindowsAccountName;
5. };

**Status**: An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d)-encoded, Boolean return value. This MUST be either 0 or 1. A value of 0 indicates success; a value of 1 indicates failure.

**Reserved**: A 32-bit signed integer that MUST be 0x00000000 and MUST be ignored on receipt.

**WindowsAccountName**: A [MapSvrUnicodeWindowsNameString (section 2.2.2.5)](#Section_2cab5dbd18cb4ce8b416ae5b23b994ba) that contains the name of the mapped Windows user or group account that MUST be in the form "DOMAIN\NAME". The length of the string MUST NOT exceed 512 bytes. Windows user account and group account name constraints are specified in [[WINUGA]](https://go.microsoft.com/fwlink/?LinkId=90568), and Windows domain naming conventions are specified in [[WINNSP]](https://go.microsoft.com/fwlink/?LinkId=90567). If **Status** does not equal 0x00000000, this value MUST be ignored.

#### windows\_account

This type is used to specify a Windows account name in [**MBCS**](#gt_8249acae-6bac-450e-9f08-1cb90ce200bd) format used to search for the corresponding [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account information. For more information, see sections [2.2.5.3](#Section_066f33191f294f649f07edb337e90d32) and [2.2.5.9](#Section_0d2556076a2647a3855bb1790221e020).

1. struct windows\_account {
2. MapSvrMBCSNameString WindowsAccountName;
3. };

**WindowsAccountName**: A [MapSvrMBCSNameString (section 2.2.2.2)](#Section_ff8f275537504d02be9ce4e3b1619b5f) that MUST contain the name of the Windows account in the "DOMAIN\NAME" format to use as the search criterion. Windows user account and group account name constraints are specified in [[WINUGA]](https://go.microsoft.com/fwlink/?LinkId=90568), and Windows domain naming conventions are specified in [[WINNSP]](https://go.microsoft.com/fwlink/?LinkId=90567). The length of the string MUST NOT exceed 256 bytes.

#### windows\_accountW

This type is used to specify a Windows account name in [**Unicode**](#gt_c305d0ab-8b94-461a-bd76-13b40cb8c4d8) format used to search for the corresponding [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account information. For more information, see sections [2.2.5.14](#Section_680ba18727d7457c9bb23b8ca9e7f259) and [2.2.5.17](#Section_d3649cf4fe6a4681b8d5879f75de6436).

1. struct windows\_accountW {
2. MapSvrUnicodeNameString WindowsAccountName;
3. };

**WindowsAccountName**: A [MapSvrUnicodeNameString (section 2.2.2.3)](#Section_dbd5ac38df0c4798a86ead90687b0ca2) that contains the name of the Windows user account to use as the search criterion. The account name MUST be in the form "DOMAIN\NAME". Windows user account and group account name constraints are specified in [[WINUGA]](https://go.microsoft.com/fwlink/?LinkId=90568), and Windows domain naming conventions are specified in [[WINNSP]](https://go.microsoft.com/fwlink/?LinkId=90567). The length of the string MUST NOT exceed 512 bytes.

#### unix\_auth

This type is used to specify [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account details returned as a result of an authentication operation on the server. For more information, see sections [2.2.5.4](#Section_c7f40a27eb2249d7a1d77e428bc61eb9) and [2.2.5.15](#Section_f80f8d46fca64f58b7e06e7f478eb121).

1. struct unix\_auth {
2. MapSvrMBCSNameString UnixAccountPassword;
3. long ID;
4. long GIDArray<MAXGIDS>;
5. };

**UnixAccountPassword**: A [MapSvrMBCSNameString (section 2.2.2.2)](#Section_ff8f275537504d02be9ce4e3b1619b5f) that contains the password of the mapped UNIX account. The length of the string MUST NOT exceed 128 bytes.

**ID**: An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d)-encoded, 32-bit signed integer that contains the UNIX [**user ID**](#gt_9c7c7fef-9933-4226-87a4-ea2a48f412c2) for the **UnixAccountPassword** that was looked up.

**GIDArray**: An array of XDR-encoded, 32-bit signed integers that contains the group IDs for the **UnixAccountPassword** that was looked up. The maximum size of this array is MAXGIDS.

#### unix\_authW

This type is used to specify [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account details returned as a result of an authentication operation on the server. For more information, see sections [2.2.5.4](#Section_c7f40a27eb2249d7a1d77e428bc61eb9) and [2.2.5.15](#Section_f80f8d46fca64f58b7e06e7f478eb121).

1. struct unix\_authW {
2. MapSvrUnicodeNameString UnixAccountPassword;
3. long ID;
4. long GIDArray<MAXGIDS>;
5. };

**UnixAccountPassword**: A [MapSvrUnicodeNameString (section 2.2.2.3)](#Section_dbd5ac38df0c4798a86ead90687b0ca2) that contains the password of the mapped UNIX account. The length of the string MUST NOT exceed 256 bytes.

**ID**: An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d)-encoded, 32-bit signed integer that contains the UNIX [**user ID**](#gt_9c7c7fef-9933-4226-87a4-ea2a48f412c2) for the **UnixAccountPassword** that was looked up.

**GIDArray**: An array of XDR-encoded, 32-bit signed integers that contains the group IDs for the **UnixAccountPassword** that was looked up. The maximum size of this array is MAXGIDS.

#### unix\_creds

This type is used to specify [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account details returned as a result of a lookup operation on the server. For more information, see sections [2.2.5.3](#Section_066f33191f294f649f07edb337e90d32), [2.2.5.9](#Section_0d2556076a2647a3855bb1790221e020), and [2.2.5.10](#Section_bb264bb5231742f181d24c705d29bf15).

1. struct unix\_creds {
2. MapSvrMBCSNameString UnixAccountName;
3. long ID;
4. long GIDArray<MAXGIDS>;
5. };

**UnixAccountName**: A [MapSvrMBCSNameString (section 2.2.2.2)](#Section_ff8f275537504d02be9ce4e3b1619b5f) that contains the name of the mapped UNIX account. The length of the string MUST NOT exceed 128 bytes.

**ID**: An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d)-encoded, 32-bit signed integer that contains the UNIX [**user ID**](#gt_9c7c7fef-9933-4226-87a4-ea2a48f412c2) for **UnixAccountName**.

**GIDArray**: An array of XDR-encoded, 32-bit signed integers that contains the group IDs for **UnixAccountName**. The maximum size of this array is MAXGIDS.

#### unix\_credsW

This type is used to specify [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account details returned as a result of a lookup operation on the server. For more information, see sections [2.2.5.14](#Section_680ba18727d7457c9bb23b8ca9e7f259), [2.2.5.17](#Section_d3649cf4fe6a4681b8d5879f75de6436), and [2.2.5.18](#Section_4f65e9e1c46b43fab3c02537e5766b3b).

1. struct unix\_credsW {
2. MapSvrUnicodeNameString UnixAccountName;
3. long ID;
4. long GIDArray<MAXGIDS>;
5. };

**UnixAccountName**: A [MapSvrUnicodeNameString (section 2.2.2.3)](#Section_dbd5ac38df0c4798a86ead90687b0ca2) that contains the name of the mapped UNIX account. The length of the string MUST NOT exceed 256 bytes.

**ID**: An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d)-encoded, 32-bit signed integer that contains the UNIX [**user ID**](#gt_9c7c7fef-9933-4226-87a4-ea2a48f412c2) for **UnixAccountName**.

**GIDArray**: An array of XDR-encoded, 32-bit signed integers that contains the group IDs for **UnixAccountName**. The maximum size of this array is MAXGIDS.

#### dump\_map\_req

This type is used to specify an input parameter to start or continue a map enumeration request to the server. For more information, see sections [2.2.5.5](#Section_50e135e64d374c85ac7d845ba95f6516), [2.2.5.7](#Section_a905c191dcce4c1187569c717a726cbf), [2.2.5.11](#Section_a06afaccc7194e83956c605013d857b0), and [2.2.5.12](#Section_66b7197697114ce899f841b062e9e07f).

1. struct dump\_map\_req {
2. long PrincipalType;
3. long MapRecordIndex;
4. };

**PrincipalType**: An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d)-encoded, 32-bit signed integer that defines the type of account mapping to enumerate. **PrincipalType** MUST be one of the following values.

|  Value |  Meaning  |
| --- | --- |
| 0x00000000 | Enumerate user account mappings. |
| 0x00000001 | Enumerate group account mappings. |

**MapRecordIndex**: An XDR-encoded, 32-bit signed integer that is an index into the set of mapping records. This MUST be set to 0 on the first call in an enumeration sequence, and to the sum of all the records returned by all preceding replies on subsequent calls in the enumeration sequence. For more information on enumeration sequences, see sections [3.1.5](#Section_dda41891285340d9a0dd853d8e275e87) and [3.2.5](#Section_d24c6377586841dd8b9786b7c0c84091).

#### sequence\_number

This type is used by the server to define a version for a set of account mappings at a given point in time. This number is changed by the server whenever any changes are made to the set of account mappings that it maintains (for more information, see section [2.2.5.6](#Section_5ec438be26d345809cd3ad4bab44c9e9)). If either of the member fields changes, the sequence\_number as a whole MUST be considered as changed.

1. struct sequence\_number {
2. long CurrentVersionTokenLowPart;
3. long CurrentVersionTokenHighPart;
4. };

**CurrentVersionTokenLowPart**: An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d)-encoded, 32-bit signed integer that MUST be either 0x00000000 or a value returned by the server from a previous call to GETCURRENTVERSIONTOKEN\_PROC or [**DUMPALLMAPSXXX\_PROC**](#gt_a85344db-a5da-41e9-a60c-36d344a282b5). For more information about **CurrentVersionTokenLowPart**, see sections [3.1.5](#Section_dda41891285340d9a0dd853d8e275e87) and [3.2.5](#Section_d24c6377586841dd8b9786b7c0c84091).

**CurrentVersionTokenHighPart**: An XDR-encoded, 32-bit signed integer that MUST be either 0x00000000 or a value returned by the server from a previous call to GETCURRENTVERSIONTOKEN\_PROC or DUMPALLMAPSXXX\_PROC. For more information about **CurrentVersionTokenHighPart**, see sections 3.1.5 and 3.2.5.

#### mapping\_record

This type is used to define a single account map when returned as an output from the map enumeration procedure. For more information, see section [2.2.3.1](#Section_985b1e0747624be3849a9fb2dc6f1a8c).

1. struct mapping\_record {
2. MapSvrMBCSNameString WindowsAccountName;
3. MapSvrMBCSNameString UnixAccountName;
4. long ID;
5. };

**WindowsAccountName**: A [MapSvrMBCSNameString (section 2.2.2.2)](#Section_ff8f275537504d02be9ce4e3b1619b5f) that contains the name of the Windows user or group account in the enumeration. The length of the string MUST NOT exceed 128 bytes. The Windows account name MUST be in the "DOMAIN\NAME" format.

**UnixAccountName**: A MapSvrMBCSNameString that contains the name of the [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) user or group account in the enumeration. The length of the string MUST NOT exceed 128 bytes.

**ID**: An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d)-encoded, 32-bit signed integer that contains the UNIX [**user ID**](#gt_9c7c7fef-9933-4226-87a4-ea2a48f412c2) or group ID for **UnixAccountName** as specified by **PrincipalType** in the request (section [2.2.5.5](#Section_50e135e64d374c85ac7d845ba95f6516)).

#### sid

This type is used to define a Windows account [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) when used as input to look up the [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account mapping details that correspond to the Windows account represented by this SID. For more information, see sections [2.2.5.10](#Section_bb264bb5231742f181d24c705d29bf15) and [2.2.5.18](#Section_4f65e9e1c46b43fab3c02537e5766b3b).

1. struct sid {
2. char SID<MAXSIDLEN>;
3. };

**SID**: An array of [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d)-encoded unsigned bytes that is a stream representation of the Windows account SID, as specified in [[MS-DTYP]](%5BMS-DTYP%5D.pdf#Section_cca2742956894a16b2b49325d93e4ba2) section 2.4.2.2. The **SubAuthority** field of the **SID** ([MS-DTYP] section 2.4.2.2) packet is a variable-length array of unsigned 32-bit [**little-endian**](#gt_079478cb-f4c5-4ce5-b72b-2144da5d2ce7) integers. The **sid** structure is an opaque data type generated by the Windows security subsystem. It is not converted to any byte-ordered network representation, and SHOULD NOT be interpreted by the User Name Mapping Protocol client or server directly; instead, it SHOULD be supplied to the underlying implementation-defined security subsystem. The maximum size of the **SID** array in the **sid** structure is MAXSIDLEN.

**Note**  Because the SID is transmitted as a raw array of bytes, the client and server MUST have identical native SID representations for user name mapping to succeed. See sections [4.9](#Section_03B1D0ED143E42CD867C08FFABA71307) and [4.17](#Section_BD6E3F9745234D31A5383FE91571B07F) for examples.

#### mapping\_recordW

This type is used to define a single account map when returned as output from the map enumeration procedure. For more information, see section [2.2.3.3](#Section_dd0096a2f16f46dca8ab0d18ad292a44).

1. struct mapping\_recordW {
2. MapSvrUnicodeNameString WindowsAccountName;
3. MapSvrUnicodeNameString UnixAccountName;
4. long ID;
5. };

**WindowsAccountName**: A [MapSvrUnicodeNameString (section 2.2.2.3)](#Section_dbd5ac38df0c4798a86ead90687b0ca2) that contains the name of the Windows user or group account in the enumeration. The length of the string MUST NOT exceed 256 bytes. The account name MUST be in the form "DOMAIN\NAME".

**UnixAccountName**: A MapSvrUnicodeNameString that contains the name of the [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) user or group account in the enumeration. The length of the string MUST NOT exceed 256 bytes.

**ID**: An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d)-encoded, 32-bit signed integer that contains the UNIX [**user ID**](#gt_9c7c7fef-9933-4226-87a4-ea2a48f412c2) or group ID for **UnixAccountName**, as specified by **PrincipalType** in the request (section [2.2.5.11](#Section_a06afaccc7194e83956c605013d857b0)).

### Non-XDR-Compliant Data Structures

There are four data structures that cannot be defined using a pure [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d) definition. Instead they are defined in terms of lower-level XDR primitives. The difference is as follows. XDR defines fixed-size arrays in terms of constants only. On the other hand, the User Name Mapping Protocol has four structures that use a dynamic value for the array size, and the layout of the fields in the User Name Mapping Protocol precludes the use of the XDR variable-sized array data type. For each of the four data types that follow, the structures are described as their standard XDR types, followed by an XDR vector that uses a dynamic size rather than a constant. This is very similar to the standard XDR variable-sized array but with a separate size value rather than one built into the array type.

See section [7](#Section_27b23be1334c49448175ec784cfcfb29) for sample code that shows how to encode and decode each of the four data structures.

#### mapping

This type is used to define a set of account maps when returned as output from the map enumeration procedure. For more information, see section [2.2.5.5](#Section_50e135e64d374c85ac7d845ba95f6516).

1. struct mapping {
2. sequence\_number Token;
3. long MappingRecordCount;
4. long TotalMappingRecordCount;
5. mapping\_record MapArray[MappingRecordCount];
6. };

**Token**: A [sequence\_number (section 2.2.2.21)](#Section_6b538aa7a21543c8876f93c409d3c5f3) that represents the current version of the data set maintained by the User Name Mapping Protocol server.

**MappingRecordCount**: An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d)-encoded, 32-bit signed integer that indicates the number of records that are returned in **MapArray**.

**TotalMappingRecordCount**: A 32-bit signed integer value that indicates the total number of mapping records of the specified **PrincipalType** (as specified in section 2.2.5.5) held by the server that are available to be enumerated.

**MapArray**: An array of account mapping records that is returned as a part of the current enumeration sequence, as specified in section [2.2.2.22](#Section_8153cbb8f0ea4c6d9a99688ed10a3c5a).

#### maps

This type is used to define a set of account maps in colon-delimited string format when returned as output from the map enumeration procedure. For more information, see section [2.2.5.7](#Section_a905c191dcce4c1187569c717a726cbf).

1. struct maps {
2. sequence\_number Token;
3. long MappingRecordCount;
4. long TotalMappingRecordCount;
5. MapSvrMBCSMapString MapArray[MappingRecordCount];
6. };

**Token**: A sequence of numbers that represent the version for the set of account maps returned in the current enumeration.

**MappingRecordCount**: An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d)-encoded, 32-bit signed integer that indicates the maximum number of records to be returned in the **MapArray** field.

**TotalMappingRecordCount**: A 32-bit signed integer value that indicates the total number of mapping records of the specified **PrincipalType** (as specified in section 2.2.5.7) held by the server that are available to be enumerated.

**MapArray**: An array of account mapping records that is returned as a part of the current enumeration sequence (as specified in section [2.2.2.6](#Section_d5e345407d2e42a3856a7ac74d3750e5)).

#### mappingW

This type is used to define a set of account maps when returned as output from the map enumeration procedure. For more information, see section [2.2.5.11](#Section_a06afaccc7194e83956c605013d857b0).

1. struct mappingW {
2. sequence\_number Token;
3. long MappingRecordCount;
4. long TotalMappingRecordCount;
5. mapping\_recordW MapArray[MappingRecordCount];
6. };

**Token**: A sequence number that represents the version for the set of account mappings that are returned in the current enumeration.

**MappingRecordCount**: An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d)-encoded, 32-bit signed integer that indicates the number of records that are returned in the **MapArray** field.

**TotalMappingRecordCount**: A 32-bit signed integer value that indicates the total number of mapping records of the specified **PrincipalType** (section 2.2.5.11) held by the server that are available to be enumerated.

**MapArray**: An array of account mapping records that is returned as a part of the current enumeration sequence. For more information, see section [2.2.2.24](#Section_d301c215b147412cb44b60e519d1b4e7).

#### mapsW

This type is used to define a set of account maps in colon-delimited string format when returned as an output from the map enumeration procedure. For more information, see section [2.2.5.12](#Section_66b7197697114ce899f841b062e9e07f).

1. struct mapsW {
2. sequence\_number Token;
3. long MappingRecordCount;
4. long TotalMappingRecordCount;
5. MapSvrUnicodeMapString MapArray[MappingRecordCount];
6. };

**Token**: A [sequence\_number (section 2.2.2.21)](#Section_6b538aa7a21543c8876f93c409d3c5f3).

**MappingRecordCount**: An [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d)-encoded, 32-bit signed integer that indicates the maximum number of records in **MapArray**.

**TotalMappingRecordCount**: A 32-bit signed integer value that indicates the total number of mapping records of the specified **PrincipalType** (section 2.2.5.12) held by the server that are available to be enumerated.

**MapArray**: An array of [MapSvrUnicodeMapString (section 2.2.2.7)](#Section_ce09d111bdc247eab853ff417401c7d6).

### Standard Failure Responses

[**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) defines a set of standard responses to requests that the User Name Mapping Protocol server is unable to service. The following tables list the set of status codes that can be returned by the User Name Mapping Protocol server.

If SUNRPC status is MSG\_ACCEPTED.

| Accept status |
| --- |
| SUCCESS |
| PROG\_UNAVAIL |
| PROG\_MISMATCH |
| PROC\_UNAVAIL |
| GARBAGE\_ARGS |
| SYSTEM\_ERR |

If SUNRPC status is MSG\_DENIED.

| Reject status | Reason rejected |
| --- | --- |
| RPC\_MISMATCH |  |
| AUTH\_ERROR | AUTH\_BADCRED |

These status codes have the following meanings:

* SUCCESS: RPC call executed successfully ([[RFC1057]](https://go.microsoft.com/fwlink/?LinkId=90265)).
* PROG\_UNAVAIL: Wrong PROGRAM\_NUMBER for the port ([RFC1057]).
* PROG\_MISMATCH: Unsupported protocol version number requested ([RFC1057]).
* PROC\_UNAVAIL: Nonexistent procedure number requested ([RFC1057]).
* GARBAGE\_ARGS: Supplied arguments illegal or otherwise not decodable ([RFC1057]).
* SYSTEM\_ERR: Errors like memory allocation failure ([[RFC1831]](https://go.microsoft.com/fwlink/?LinkId=90295)).
* RPC\_MISMATCH: Invalid SUNRPC version number ([RFC1057]).
* AUTH\_ERROR: Remote cannot authenticate caller ([RFC1057]).
* AUTH\_BADCRED: Bad credentials in RPC call ([RFC1057]).[<5>](#Appendix_A_5" \o "Product behavior note 5)

### User Name Mapping Protocol Messages

The User Name Mapping Protocol procedure messages are defined in the [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) request and response headers, as specified in [[RFC1057]](https://go.microsoft.com/fwlink/?LinkId=90265) section 8. The following table lists the procedure messages in procedure number order.

| Procedure name | Procedure number | Version |
| --- | --- | --- |
| MAPPROC\_NULL | 0 | 1, 2 |
| GETWINDOWSCREDSFROMUNIXUSERNAME\_PROC | 1 | 1, 2 |
| GETUNIXCREDSFROMNTUSERNAME\_PROC | 2 | 1, 2 |
| AUTHUSINGUNIXCREDS\_PROC | 3 | 1, 2 |
| DUMPALLMAPS\_PROC | 4 | 1, 2 |
| GETCURRENTVERSIONTOKEN\_PROC | 5 | 1, 2 |
| DUMPALLMAPSEX\_PROC | 6 | 1, 2 |
| GETWINDOWSGROUPFROMUNIXGROUPNAME\_PROC | 7 | 1, 2 |
| GETUNIXCREDSFROMNTGROUPNAME\_PROC | 8 | 1, 2 |
| GETUNIXCREDSFROMNTUSERSID\_PROC | 9 | 2 |
| DUMPALLMAPSW\_PROC | 10 | 2 |
| DUMPALLMAPSEXW\_PROC | 11 | 2 |
| GETWINDOWSUSERFROMUNIXUSERNAMEW\_PROC | 12 | 2 |
| GETUNIXCREDSFROMNTUSERNAMEW\_PROC | 13 | 2 |
| AUTHUSINGUNIXCREDSW\_PROC | 14 | 2 |
| GETWINDOWSGROUPFROMUNIXGROUPNAMEW\_PROC | 15 | 2 |
| GETUNIXCREDSFROMNTGROUPNAMEW\_PROC | 16 | 2 |
| GETUNIXCREDSFROMNTUSERSIDW\_PROC | 17 | 2 |

#### MAPPROC\_NULL (PROC 0)

A null procedure that is used for service discovery as specified in [[RFC1057]](https://go.microsoft.com/fwlink/?LinkId=90265) section A.2.

1. void
2. MAPPROC\_NULL(
3. void
4. );

This procedure requires no arguments, and a successful reply MUST contain no data other than a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) reply status of MSG\_ACCEPTED, as specified in [RFC1057].

The typical use of a null procedure is for the clients to discover whether the service is started and available. This procedure has a procedure number equal to 0.

#### GETWINDOWSCREDSFROMUNIXUSERNAME\_PROC (PROC 1)

A request to fetch the mapped Windows user account name for a specified [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) user name and/or UNIX user.

1. windows\_creds
2. GETWINDOWSCREDSFROMUNIXUSERNAME\_PROC(
3. unix\_account UnixUser
4. );

*UnixUser*: The UNIX user to search for, using the account name and/or UID as the search criteria, as specified by the value for SearchOption.

**Return Value:** A [windows\_creds](#Section_ae9562678a7f4567866196b64aca3367) record that contains the mapped Windows user account details for the specified UNIX user. Whenever the lookup request for a specified UNIX account succeeds or fails to find a corresponding Windows account map, the User Name Mapping Protocol server MUST return a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) status of MSG\_ACCEPTED with an accept status of SUCCESS. The actual success or failure of the request MUST be set in the Status member of the returned structure. Status is a Boolean value, with 0 indicating a successful lookup request and 1 indicating a failed lookup request.

#### GETUNIXCREDSFROMNTUSERNAME\_PROC (PROC 2)

A request to fetch the mapped [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) user account details for a specified Windows user account name.

1. unix\_creds
2. GETUNIXCREDSFROMNTUSERNAME\_PROC(
3. windows\_account WindowsUserAccountName
4. );

*WindowsUserAccountName*: The Windows user to use for the account name as the search criterion.

**Return Value:** A [unix\_creds](#Section_d63527bebf8a44cb81a7ccfbeb7e074e) record containing the mapped UNIX user account details for the specified Windows account name. Whenever the lookup request for a specified Windows account fails to find a corresponding UNIX account map, the User Name Mapping Protocol server MUST return a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) status of MSG\_ACCEPTED with an accept status of SUCCESS. It MUST also return a zero-length string in the **UnixAccountName** member of the returned structure.

#### AUTHUSINGUNIXCREDS\_PROC (PROC 3)

A request to fetch the [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account details for a given UNIX user name and password.

This procedure is typically used by clients that are doing a simple authentication by providing a user name and password. The password string is the string returned by a call to the crypt() API using the user's cleartext password, as described in section 3 of the System Interfaces Volume (XSH) of [[IEEE1003.1]](https://go.microsoft.com/fwlink/?LinkId=89897).

1. unix\_auth
2. AUTHUSINGUNIXCREDS\_PROC(
3. unix\_user\_auth UnixUserAuth
4. );

*UnixUserAuth*: UNIX user name and password to use as the search criteria.[<6>](#Appendix_A_6" \o "Product behavior note 6)

**Return Value:** A [unix\_auth](#Section_1fa00f3da1fe45949c2c7530c17b20c7) record that contains the mapped UNIX user account details for the specified UNIX account. Whenever the lookup request for a specified UNIX account fails to find a corresponding UNIX account map, the User Name Mapping Protocol server MUST return a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) status of MSG\_ACCEPTED with an accept status of SUCCESS. It MUST also return a zero-length string in the **UnixAccountPassword** member of the returned structure.

#### DUMPALLMAPS\_PROC (PROC 4)

A request to enumerate all account mappings held by the service.

1. mapping
2. DUMPALLMAPS\_PROC(
3. dump\_map\_req EnumCursor
4. );

*EnumCursor*: A **PrincipalType** and index to start or continue an enumeration.

**Return Value:** A mapping type that describes an array of zero or more [mapping\_record](#Section_8153cbb8f0ea4c6d9a99688ed10a3c5a) types. Whenever the enumeration request fails to find any records to either begin or continue the enumeration, the User Name Mapping Protocol server MUST return a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) status of MSG\_ACCEPTED with an accept status of SUCCESS, and MUST return 0 in the **MappingRecordCount** field. It MUST also return a zero-length set of mapping\_record types in the **MapArray** member of the returned structure. The User Name Mapping Protocol server MUST also return current values for the server [sequence\_number](#Section_6b538aa7a21543c8876f93c409d3c5f3) in the **Token** field and the total mapping record count for the specified enumeration in the **TotalMappingRecordCount** field of the returned structure.

#### GETCURRENTVERSIONTOKEN\_PROC (PROC 5)

A request for the current account-mapping sequence number for the set of mapping records held by the server. This procedure is used by clients to check whether any [**map**](#gt_173464f7-c0e8-4181-941a-f7df6725be5a) records changed since the last enumeration by the client.

1. sequence\_number
2. GETCURRENTVERSIONTOKEN\_PROC(
3. sequence\_number SequenceNumber
4. );

*SequenceNumber*: A data structure that contains two 32-bit signed integers. *SequenceNumber* MUST contain either 0x00000000 for each member field or a value returned by the server from a previous call to GETCURRENTVERSIONTOKEN\_PROC or [**DUMPALLMAPSXXX\_PROC**](#gt_a85344db-a5da-41e9-a60c-36d344a282b5). For more information, see sections [3.1.5](#Section_dda41891285340d9a0dd853d8e275e87) and [3.2.5](#Section_d24c6377586841dd8b9786b7c0c84091).

**Return Value:** The User Name Mapping Protocol server MUST return a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) status of MSG\_ACCEPTED with an accept status of SUCCESS. It MUST also return a [sequence\_number](#Section_6b538aa7a21543c8876f93c409d3c5f3) structure with the current sequence number value for the set of mapping records held by the server.

#### DUMPALLMAPSEX\_PROC (PROC 6)

A request to enumerate all account mappings held by the service.

1. maps
2. DUMPALLMAPSEX\_PROC(
3. dump\_map\_req EnumCursor
4. );

*EnumCursor*: A **PrincipalType** and index to start or continue an enumeration.

**Return Value:** A [maps](#Section_07e03281db04473588930136b435ddb4) type record that describes an array of zero or more [MapSvrMBCSMapString](#Section_d5e345407d2e42a3856a7ac74d3750e5) types. Whenever the enumeration request fails to find any records to either begin or continue the enumeration, the User Name Mapping Protocol server MUST return a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) status of MSG\_ACCEPTED with an accept status of SUCCESS, and MUST return 0 in the **MappingRecordCount** field. It MUST also return a zero-length set of MapSvrMBCSMapString types in the **MapArray** member of the returned structure.

The User Name Mapping Protocol server MUST also return current values for the server [sequence\_number](#Section_6b538aa7a21543c8876f93c409d3c5f3) in the **Token** field and the total mapping record count for the specified enumeration in the **TotalMappingRecordCount** field of the returned structure.

#### GETWINDOWSGROUPFROMUNIXGROUPNAME\_PROC (PROC 7)

A request to fetch the Windows group account information that corresponds to a [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) group name.

1. windows\_creds
2. GETWINDOWSGROUPFROMUNIXGROUPNAME\_PROC(
3. unix\_account UnixGroupAccount
4. );

*UnixGroupAccount*: A UNIX group to search for, using the account name and/or [**GID**](#gt_323122fc-2cf0-42be-98c6-cb492663c44b) as the search criteria, as specified by the value for **SearchOption**.

**Return Value:** A [windows\_creds](#Section_ae9562678a7f4567866196b64aca3367) record that contains the mapped Windows group account details for the specified UNIX group. Whenever the lookup request for a specified UNIX account succeeds or fails to find a corresponding Windows account [**map**](#gt_173464f7-c0e8-4181-941a-f7df6725be5a), the User Name Mapping Protocol server MUST return a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) status of MSG\_ACCEPTED with an accept status of SUCCESS. The actual success or failure of the request MUST be set in the **Status** member of the returned structure. **Status** is a Boolean value, with 0 indicating a successful lookup request and 1 indicating a failed lookup request.

#### GETUNIXCREDSFROMNTGROUPNAME\_PROC (PROC 8)

A request to fetch the [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) group account information that corresponds to a Windows group name.

1. unix\_creds
2. GETUNIXCREDSFROMNTGROUPNAME\_PROC(
3. windows\_account WindowsGroupAccountName
4. );

*WindowsGroupAccountName*: A Windows group to use for the account name as the search criteria.

**Return Value:** A [unix\_creds](#Section_d63527bebf8a44cb81a7ccfbeb7e074e) record that contains the mapped UNIX group account details for the specified Windows account name. Whenever the lookup request for a specified Windows account fails to find a corresponding UNIX account [**map**](#gt_173464f7-c0e8-4181-941a-f7df6725be5a), the User Name Mapping Protocol server MUST return a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) status of MSG\_ACCEPTED with an accept status of SUCCESS. It MUST also return a zero-length string in the **UnixAccountName** member of the returned structure.

#### GETUNIXCREDSFROMNTUSERSID\_PROC (PROC 9)

A request for the [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account information that corresponds to the Windows account specified by the [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d).

1. unix\_creds
2. GETUNIXCREDSFROMNTUSERSID\_PROC(
3. sid SID
4. );

*SID*: A Windows SID to use as the search criteria.

**Return Value:** A [unix\_creds](#Section_d63527bebf8a44cb81a7ccfbeb7e074e) record that contains the mapped UNIX account details for the specified Windows SID. Whenever the lookup request for a specified Windows SID fails to find a corresponding UNIX account map, the User Name Mapping Protocol server MUST return a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) status of MSG\_ACCEPTED with an accept status of SUCCESS. It MUST also return a zero-length string in the **UnixAccountName** member of the returned structure.

#### DUMPALLMAPSW\_PROC (PROC 10)

This procedure is the [**wide character**](#gt_30c30de3-1d00-4d0d-9109-fcc0094fc95a) counterpart of [DUMPALLMAPS\_PROC](#Section_50e135e64d374c85ac7d845ba95f6516). The request and response packets are identical to DUMPALLMAPS\_PROC, except that the return value is a [mappingW](#Section_dd0096a2f16f46dca8ab0d18ad292a44) data type instead of a [mapping](#Section_985b1e0747624be3849a9fb2dc6f1a8c) data type. For example, the [MapSvrMBCSNameString](#Section_ff8f275537504d02be9ce4e3b1619b5f) data type is replaced with a [MapSvrUnicodeNameString](#Section_dbd5ac38df0c4798a86ead90687b0ca2) type in the byte stream.

1. mappingW
2. DUMPALLMAPSW\_PROC(
3. dump\_map\_req EnumCursor
4. );

*EnumCursor*: A **PrincipalType** and index to start or continue an enumeration.

**Return Value:** A mappingW type that describes an array of zero or more [mapping\_recordW](#Section_d301c215b147412cb44b60e519d1b4e7) types. Whenever the enumeration request fails to find any records to either begin or continue the enumeration, the User Name Mapping Protocol server MUST return a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) status of MSG\_ACCEPTED with an accept status of SUCCESS, and MUST return 0 in the **MappingRecordCount** field. It MUST also return a zero-length set of mapping\_recordW types in the **MapArray** member of the returned structure. The User Name Mapping Protocol server MUST also return current values for the server [sequence\_number](#Section_6b538aa7a21543c8876f93c409d3c5f3) in the **Token** field, and the total mapping record count for the specified enumeration in the **TotalMappingRecordCount** field of the returned structure.

#### DUMPALLMAPSEXW\_PROC (PROC 11)

This procedure is the [**wide character**](#gt_30c30de3-1d00-4d0d-9109-fcc0094fc95a) counterpart of [DUMPALLMAPSEX\_PROC](#Section_a905c191dcce4c1187569c717a726cbf). The request and response packets are identical to DUMPALLMAPSEX\_PROC, except that the [MapSvrMBCSMapString](#Section_d5e345407d2e42a3856a7ac74d3750e5) data type is replaced with a [MapSvrUnicodeMapString](#Section_ce09d111bdc247eab853ff417401c7d6) type in the byte stream.

1. mapsW
2. DUMPALLMAPSEXW\_PROC(
3. dump\_map\_req EnumCursor
4. );

*EnumCursor*: A **PrincipalType** and index to start or continue an enumeration.

**Return Value:** A [mapsW](#Section_003667ced8bb4e5eafb6a73e3340f67a) type record that describes an array of zero or more MapSvrUnicodeMapString types. Whenever the enumeration request fails to find any records to either begin or continue the enumeration, the User Name Mapping Protocol server MUST return a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) status of MSG\_ACCEPTED with an accept status of SUCCESS, and MUST return 0 in the **MappingRecordCount** field. It MUST also return a zero-length set of MapSvrUnicodeMapString types in the **MapArray** member of the returned structure. The User Name Mapping Protocol server MUST also return current values for the server [sequence\_number](#Section_6b538aa7a21543c8876f93c409d3c5f3) in the **Token** field, and the total mapping record count for the specified enumeration in the **TotalMappingRecordCount** field of the returned structure.

#### GETWINDOWSUSERFROMUNIXUSERNAMEW\_PROC (PROC 12)

This procedure is the [**wide character**](#gt_30c30de3-1d00-4d0d-9109-fcc0094fc95a) counterpart to [GETWINDOWSCREDSFROMUNIXUSERNAME\_PROC](#Section_887248fa50f84268a47d27583ff97cf8). The request and response packets are identical to GETWINDOWSCREDSFROMUNIXUSERNAME\_PROC, except that the [MapSvrMBCSNameString](#Section_ff8f275537504d02be9ce4e3b1619b5f) data type is replaced with a [MapSvrUnicodeNameString](#Section_dbd5ac38df0c4798a86ead90687b0ca2) type in the byte stream.

1. windows\_credsW
2. GETWINDOWSUSERFROMUNIXUSERNAMEW\_PROC(
3. unix\_accountW UnixUser
4. );

*UnixUser*: A [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account to search for, using the account name and/or UID as the search criteria, as specified by the value for **SearchOption**.

**Return Value:** A [windows\_credsW](#Section_165f85c0c7f64ddbadc488e2ed0e772f) record that contains the mapped Windows user account details for the specified UNIX user account. Whenever the lookup request for a specified UNIX account succeeds or fails to find a corresponding Windows account [**map**](#gt_173464f7-c0e8-4181-941a-f7df6725be5a), the User Name Mapping Protocol server MUST return a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) status of MSG\_ACCEPTED with an accept status of SUCCESS. The actual success or failure of the request MUST be set in the **Status** member of the returned structure. **Status** is a Boolean value, with 0 indicating a successful lookup request and 1 indicating a failed lookup request.

#### GETUNIXCREDSFROMNTUSERNAMEW\_PROC (PROC 13)

This procedure is the [**wide character**](#gt_30c30de3-1d00-4d0d-9109-fcc0094fc95a) counterpart to [GETUNIXCREDSFROMNTUSERNAME\_PROC](#Section_066f33191f294f649f07edb337e90d32). The request and response packets are identical to GETUNIXCREDSFROMNTUSERNAME\_PROC, except that the [MapSvrMBCSNameString](#Section_ff8f275537504d02be9ce4e3b1619b5f) data type is replaced with a [MapSvrUnicodeNameString](#Section_dbd5ac38df0c4798a86ead90687b0ca2) type in the byte stream.

1. unix\_credsW
2. GETUNIXCREDSFROMNTUSERNAMEW\_PROC(
3. windows\_accountW WindowsUserAccountName
4. );

*WindowsUserAccountName*: A Windows account to use for the account name as the search criterion.

**Return Value:** A [unix\_credsW](#Section_af4ae7269e04495f92fab22931ebb979) record that contains the mapped [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) user account details for the specified Windows account name. Whenever the lookup request for a specified Windows account fails to find a corresponding UNIX account map, the User Name Mapping Protocol server MUST return a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) status of MSG\_ACCEPTED with an accept status of SUCCESS. It MUST also return a zero-length string in the **UnixAccountName** member of the returned structure.

#### AUTHUSINGUNIXCREDSW\_PROC (PROC 14)

This procedure is the [**wide character**](#gt_30c30de3-1d00-4d0d-9109-fcc0094fc95a) counterpart to [AUTHUSINGUNIXCREDS\_PROC](#Section_c7f40a27eb2249d7a1d77e428bc61eb9). The request and response packets are identical to AUTHUSINGUNIXCREDS\_PROC, except that the [MapSvrMBCSNameString](#Section_ff8f275537504d02be9ce4e3b1619b5f) data type is replaced with a [MapSvrUnicodeNameString](#Section_dbd5ac38df0c4798a86ead90687b0ca2) type in the byte stream.

1. unix\_authW
2. AUTHUSINGUNIXCREDSW\_PROC(
3. unix\_user\_authW UnixUserAuth
4. );

*UnixUserAuth*: A [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) user name and password to use as the search criteria.[<7>](#Appendix_A_7" \o "Product behavior note 7)

**Return Value:** A [unix\_authW](#Section_0085175ecf324007b4ee997f29e807b3) record that contains the mapped UNIX user account details for the specified UNIX account. Whenever the lookup request for a specified UNIX account fails to find a corresponding UNIX account map, the User Name Mapping Protocol server MUST return a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) status of MSG\_ACCEPTED with an accept status of SUCCESS. It MUST also return a zero-length string in the **UnixAccountPassword** member of the returned structure.

#### GETWINDOWSGROUPFROMUNIXGROUPNAMEW\_PROC (PROC 15)

This procedure is the [**wide character**](#gt_30c30de3-1d00-4d0d-9109-fcc0094fc95a) counterpart to [GETWINDOWSGROUPFROMUNIXGROUPNAME\_PROC](#Section_ef1fd6724c4646d0b3b779c2fa8e62e9). The request and response packets are identical to GETWINDOWSGROUPFROMUNIXGROUPNAME\_PROC, except that the [MapSvrMBCSNameString](#Section_ff8f275537504d02be9ce4e3b1619b5f) data type is replaced with a [MapSvrUnicodeNameString](#Section_dbd5ac38df0c4798a86ead90687b0ca2) type in the byte stream.

1. windows\_credsW
2. GETWINDOWSGROUPFROMUNIXGROUPNAMEW\_PROC(
3. unix\_accountW UnixGroupAccount
4. );

*UnixGroupAccount*: A [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) group to search for, using the account name and/or [**GID**](#gt_323122fc-2cf0-42be-98c6-cb492663c44b) as the search criteria, as specified by the value for **SearchOption**.

**Return Value:** A [windows\_credsW](#Section_165f85c0c7f64ddbadc488e2ed0e772f) record that contains the mapped Windows group account details for the specified UNIX group. Whenever the lookup request for a specified UNIX account succeeds or fails to find a corresponding Windows account map, the User Name Mapping Protocol server MUST return a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) status of MSG\_ACCEPTED with an accept status of SUCCESS. The actual success or failure of the request MUST be set in the **Status** member of the returned structure. **Status** is a Boolean value, with 0 indicating a successful lookup request and 1 indicating a failed lookup request.

#### GETUNIXCREDSFROMNTGROUPNAMEW\_PROC (PROC 16)

This procedure is the [**wide character**](#gt_30c30de3-1d00-4d0d-9109-fcc0094fc95a) counterpart to [GETUNIXCREDSFROMNTGROUPNAME\_PROC](#Section_0d2556076a2647a3855bb1790221e020). The request and response packets are identical to GETUNIXCREDSFROMNTGROUPNAME\_PROC, except that the [MapSvrMBCSNameString](#Section_ff8f275537504d02be9ce4e3b1619b5f) data type is replaced with a [MapSvrUnicodeNameString](#Section_dbd5ac38df0c4798a86ead90687b0ca2) type in the byte stream.

1. unix\_credsW
2. GETUNIXCREDSFROMNTGROUPNAMEW\_PROC(
3. windows\_accountW WindowsGroupAccountName
4. );

*WindowsGroupAccountName*: A Windows group to use as the account name in the search criteria.

**Return Value:** A [unix\_credsW](#Section_af4ae7269e04495f92fab22931ebb979) record that contains the mapped [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) group account details for the specified Windows account name. Whenever the lookup request for a specified Windows account fails to find a corresponding UNIX account map, the User Name Mapping Protocol server MUST return a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) status of MSG\_ACCEPTED with an accept status of SUCCESS. It MUST also return a zero-length string in the **UnixAccountName** member of the returned structure.

#### GETUNIXCREDSFROMNTUSERSIDW\_PROC (PROC 17)

This procedure is the [**wide character**](#gt_30c30de3-1d00-4d0d-9109-fcc0094fc95a) counterpart to [GETUNIXCREDSFROMNTUSERSID\_PROC](#Section_bb264bb5231742f181d24c705d29bf15). The request and response packets are identical to GETUNIXCREDSFROMNTUSERSID\_PROC, except that the [MapSvrMBCSNameString](#Section_ff8f275537504d02be9ce4e3b1619b5f) data type is replaced with a [MapSvrUnicodeNameString](#Section_dbd5ac38df0c4798a86ead90687b0ca2) type in the byte stream.

1. unix\_credsW
2. GETUNIXCREDSFROMNTUSERSIDW\_PROC(
3. sid SID
4. );

*SID*: A Windows [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) to use as the search criteria.

**Return Value:** A [unix\_credsW](#Section_af4ae7269e04495f92fab22931ebb979) record that contains the mapped [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account details for the specified Windows SID. Whenever the lookup request for a specified Windows SID fails to find a corresponding UNIX account map, the User Name Mapping Protocol server MUST return a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) status of MSG\_ACCEPTED with an accept status of SUCCESS. It MUST also return a zero-length string in the **UnixAccountName** member of the returned structure.

# Protocol Details

With the exception of the [**DUMPALLMAPSXXX\_PROC**](#gt_a85344db-a5da-41e9-a60c-36d344a282b5) procedures, requests sent by the User Name Mapping Protocol client generate a single response from the User Name Mapping Protocol server. There is no predetermined sequencing.

The DUMPALLMAPSXXX\_PROC procedures are used to enumerate some or all of the mapping records held by the User Name Mapping Protocol server. All the DUMPALLMAPSXXX\_PROC procedures follow the same sequencing rules, as defined in the following sections. The sequence can be restricted to a single request-response pair, or it can extend over many request-response pairs, depending on the number of maps available on the User Name Mapping Protocol server and the requirements of the User Name Mapping Protocol client.

Each enumeration sequence is independent of other individual requests or enumeration sequences between the User Name Mapping Protocol client and server. Therefore, multiple enumerations (from the same or different clients) for [**user map**](#gt_6f6cefa4-0c32-4524-a267-2ce2cb29d7e9) and [**group map**](#gt_5501e0d8-1bdc-464c-8c40-c78de7b96734) can proceed in parallel without any interference.

## Client Details

### Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model, as long as their external behavior is consistent with that described in this document.

Clients of the User Name Mapping Protocol can maintain copies of user mappings (and group mappings) enumerated from the server. Clients can use the [**DUMPALLMAPSXXX\_PROC**](#gt_a85344db-a5da-41e9-a60c-36d344a282b5) procedures to enumerate all individual maps from the server. The server treats account mappings as an unordered array of mapping records of total count equal to **TotalMappingRecordCount**, as explained in sections [2.2.3.1](#Section_985b1e0747624be3849a9fb2dc6f1a8c) and [2.2.3.3](#Section_dd0096a2f16f46dca8ab0d18ad292a44). The index of records begins at zero, and **MappingRecordCount** indicates the number of [**map**](#gt_173464f7-c0e8-4181-941a-f7df6725be5a) records returned by the server in the current RPC response packet.

Clients can cache **CurrentVersionTokenHighPart** and **CurrentVersionTokenLowPart** values returned by the DUMPALLMAPSXXX\_PROC response to implement cache consistency. Cache consistency is implemented on clients by periodically polling the server's [GETCURRENTVERSIONTOKEN\_PROC](#Section_5ec438be26d345809cd3ad4bab44c9e9) procedure to know when to refresh their locally cached copies of mappings.

As an alternative to the enumeration request (DUMPALLMAPSXXX\_PROC), the clients can cache the results of individual account lookup requests and use GETCURRENTVERSIONTOKEN\_PROC to know when to refresh their locally cached copies of mappings.

Clients of the User Name Mapping Protocol are at liberty to implement caching and persistence in any way they please. The User Name Mapping Protocol server functions as a read-only lookup service of account mappings.

The following figure shows the data model for a client of the User Name Mapping Protocol.



Figure 1: User Name Mapping Protocol data model: Client

There are three elements in the model: MapCache, MapRecord, and CurrentVersionToken.

**MapCache:** The MapCache element models the information that the client has collected from the server by enumerating maps using the DUMPALLMAPSXXX\_PROC. The MapCache element contains a list (or array) of MapRecord elements, each of which describes the mapping between a Windows and a [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account.

**MapRecord:** The MapRecord element models the information for a single Windows-to-UNIX user account mapping or group account mapping. It contains the UNIX account name and UID, a [**GID**](#gt_323122fc-2cf0-42be-98c6-cb492663c44b), and the supplementary GID details that correspond to a Windows account name and [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).

**CurrentVersionToken:** This element models the version of the cache as a whole. This element is guaranteed by the server to be different for different versions of the MapCache. Clients can use this element to implement cache consistency with respect to the server by periodically polling this token by using the GETCURRENTVERSIONTOKEN\_PROC procedure.

### Timers

There are no timers in the User Name Mapping Protocol beyond those used by [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600).

### Initialization

None.

### Higher-Layer Triggered Events

None.

### Message Processing Events and Sequencing Rules

The User Name Mapping Protocol allows a User Name Mapping Protocol client to retrieve a complete set of account mappings from the server and to maintain a copy of these mappings in a local cache. The client uses a combination of the [**DUMPALLMAPSXXX\_PROC**](#gt_a85344db-a5da-41e9-a60c-36d344a282b5) and [GETCURRENTVERSIONTOKEN\_PROC](#Section_5ec438be26d345809cd3ad4bab44c9e9) procedure calls to retrieve the account mappings and to check for updates to the account mappings in the server, respectively. The DUMPALLMAPSXXX\_PROC procedure that is chosen is determined by the type of information that the User Name Mapping Protocol client chooses to cache.

All procedures other than DUMPALLMAPSXXX\_PROC are self-contained in that they do not require any other procedures to be sequenced in order to complete successfully. The User Name Mapping Protocol client does not need to maintain any state to implement sequencing across procedure calls.

For all procedures, the processing rules for a server-returned response packet are specified in [[RFC1057]](https://go.microsoft.com/fwlink/?LinkId=90265) section 8. The client MUST interpret server procedure response status of MSG\_ACCEPTED or MSG\_DENIED according to those rules.

#### Making the Initial Account Mapping Request to the Server

The sequence begins when a User Name Mapping Protocol client sends a [**DUMPALLMAPSXXX\_PROC**](#gt_a85344db-a5da-41e9-a60c-36d344a282b5) procedure request to the server, with the **MapRecordIndex** field equal to 0 to indicate the start of a new enumeration sequence and the **PrincipalType** field equal to the record type to be returned.

#### Processing the Account Mapping Response from the Server

If the [**DUMPALLMAPSXXX\_PROC**](#gt_a85344db-a5da-41e9-a60c-36d344a282b5) response from the server indicates success and the returned value of **MappingRecordCount** is less than the returned value of **TotalMappingRecordCount**, the client proceeds to section [3.1.5.3](#Section_c5b938b777984547ba267c687c033b2c); the enumeration of account mappings returned from the server is incomplete and there are more records to retrieve.

Otherwise, the enumeration returned is complete if the response indicates success. The client MAY send another DUMPALLMAPSXXX\_PROC request to the server if the response indicates failure.

#### Making Further Account Mapping Requests to the Server

The User Name Mapping Protocol client continues to make further [**DUMPALLMAPSXXX\_PROC**](#gt_a85344db-a5da-41e9-a60c-36d344a282b5) requests, each time increasing the value of **MapRecordIndex** to the total number of map records returned by the server so far for this enumeration. For example, if the first reply returned 15 records, and the second reply returned 12 records, the third request in the sequence sets the **MapRecordIndex** to 27 (15 + 12). The User Name Mapping Protocol client continues to make requests until there are no more account mappings to retrieve from the server. This is indicated by a DUMPALLMAPSXXX\_PROC reply that contains zero records (**MappingRecordCount** is 0), or if the next DUMPALLMAPSXXX\_PROC request would set **MapRecordIndex** to **TotalMappingRecordCount**. **TotalMappingRecordCount** is returned in the server DUMPALLMAPSXXX\_PROC response.

If at any point the values of **CurrentVersionTokenHighPart**, **CurrentVersionTokenLowPart**, or **TotalMappingRecordCount** returned by the server in the DUMPALLMAPSXXX\_PROC response change from the initial values returned when **MapRecordIndex** was set to 0 in the DUMPALLMAPSXXX\_PROC request, the current enumeration MUST be abandoned and restarted with a new DUMPALLMAPSXXX\_PROC request (**MapRecordIndex** equal to 0).

#### Polling for Cache Consistency

The User Name Mapping Protocol client uses [GETCURRENTVERSIONTOKEN\_PROC](#Section_5ec438be26d345809cd3ad4bab44c9e9) to periodically check the server for cache consistency. Whenever any of the user or group account mappings on the server change, the tokens returned in the response to GETCURRENTVERSIONTOKEN\_PROC are different, at which point the client MUST discard its cached copy of all the mappings in their entirety and enumerate the new set of mappings from the server.

If **CurrentVersionTokenHighPart** and **CurrentVersionTokenLowPart** in the GETCURRENTVERSIONTOKEN\_PROC reply are the same as those from the previous enumeration, there have been no changes to any map records, and any cache of map records being maintained by the User Name Mapping Protocol client is still valid. If either **CurrentVersionTokenHighPart** or **CurrentVersionTokenLowPart** in the GETCURRENTVERSIONTOKEN\_PROC reply differs from those returned by the previous enumeration, the mapping records have been updated, and the User Name Mapping Protocol client MUST consider the local cached copies of the mapping records as out of date and MUST repeat the enumeration to get the updated set of mapping records.

### Timer Events

None.

### Local Events

None.

## Server Details

### Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model, as long as their external behavior is consistent with that described in this document.

The User Name Mapping Protocol server maintains a database of account mappings and provides procedures for enumeration of these account mappings. The server maintains a unique 64-bit sequence number that is initialized at server startup and changed whenever the database of maps is updated.

The User Name Mapping Protocol server returns the 64-bit sequence number to the clients to allow them to implement a polling-based cache consistency scheme that times out locally cached copies of account mappings on the client.

The following figure shows the data model for the User Name Mapping Protocol server.



Figure 2: User Name Mapping Protocol data model: Server

There are three elements in the model: MapDatabase, MapRecord, and CurrentVersionToken.

**MapDatabase:** The MapDatabase element models a nonvolatile store of mapping information between Windows and [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) accounts. This element contains a set of MapRecord elements and a CurrentVersionToken element.

**MapRecord:** The MapRecord element models the information for a single Windows-to-UNIX user account mapping or group account mapping. It contains the UNIX account name and UID, a [**GID**](#gt_323122fc-2cf0-42be-98c6-cb492663c44b), and the supplementary GID details that correspond to a Windows account name and [**domain**](#gt_b0276eb2-4e65-4cf1-a718-e0920a614aca).

**CurrentVersionToken:** This element models the version of the MapDatabase as a whole. This element MUST be guaranteed by the server to be unique following each update to the MapDatabase.

### Timers

None.

### Initialization

None.

### Higher-Layer Triggered Events

None.

### Message Processing Events and Sequencing Rules

#### Processing for All Procedures

The User Name Mapping Protocol server performs a simple lookup or enumeration service on behalf of clients. As described in section [3.2.1](#Section_951ed3deacce4d4aa2be27c1b4428c06), the server maintains a set of current mappings that it traverses to answer queries by clients. For each lookup procedure from the client, the User Name Mapping Protocol server queries the persistent data store of account mappings and returns details of the located map, if found.

The [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) response packet generated by the User Name Mapping Protocol server adheres to the rules indicated in [[RFC1057]](https://go.microsoft.com/fwlink/?LinkId=90265) section 8. Whenever a well-formed SUNRPC request is received, the body of the response packet MUST have a status of MSG\_ACCEPTED to indicate a successful receipt of the packet.[<8>](#Appendix_A_8" \o "Product behavior note 8)

The server MUST return an error of SUNRPC PROG\_MISMATCH whenever the client requests a program version other than 1 or 2.

In all cases where the server fails to decode the lookup or enumeration procedure request arguments, it MUST return a response error value of GARBAGE\_ARGS.

In all cases where the lookup or enumeration request succeeds, the server MUST return a SUCCESS status in the reply body and encode the procedure-specific return values according to the [**XDR**](#gt_6198c2ee-0ffa-490d-b296-08c8a428b93d) rules defined in [[RFC4506]](https://go.microsoft.com/fwlink/?LinkId=90478).

#### Processing of DUMPALLMAPSXXX\_PROC Request and GETCURRENTVERSIONTOKEN\_PROC Request

##### Processing the Initial Account Mapping Request from the Client

The User Name Mapping Protocol server replies to the [**DUMPALLMAPSXXX\_PROC**](#gt_a85344db-a5da-41e9-a60c-36d344a282b5) request with a two-part version token (**CurrentVersionTokenHighPart** and **CurrentVersionTokenLowPart**), a count of the number of maps in the reply (**MappingRecordCount**), the total number of maps available on the server (**TotalMappingRecordCount**), and a list of **MappingRecordCount** mapping records that begin at the **MapRecordIndex** index equal to 0. The number of account mapping records returned by the server to the client is implementation specific.[<9>](#Appendix_A_9" \o "Product behavior note 9)[<10>](#Appendix_A_10" \o "Product behavior note 10)

##### Processing Further Account Mapping Requests from the Client

The User Name Mapping Protocol server replies to the [**DUMPALLMAPSXXX\_PROC**](#gt_a85344db-a5da-41e9-a60c-36d344a282b5) request with the next set of mapping records, starting with the map record at the **MapRecordIndex** index requested. If the **MapRecordIndex** requested is out of bounds of the **TotalMappingRecordCount** number of account mappings stored on the server, **MappingRecordCount** MUST be returned with a value of 0, and no records are returned. The number of account mapping records returned by the server to the client is implementation specific. For example, the server might limit the number of mappings returned to the amount of data that can fit in a single [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) packet of a chosen maximum size.[<11>](#Appendix_A_11" \o "Product behavior note 11)[<12>](#Appendix_A_12" \o "Product behavior note 12)[<13>](#Appendix_A_13" \o "Product behavior note 13)

##### Processing the Client Account Mapping Cache Refresh

The User Name Mapping Protocol server replies with **CurrentVersionTokenHighPart** and **CurrentVersionTokenLowPart** in the [GETCURRENTVERSIONTOKEN\_PROC](#Section_5ec438be26d345809cd3ad4bab44c9e9) reply set to an implementation-specific value. If the account mappings have been changed since a client's previous GETCURRENTVERSIONTOKEN\_PROC or [**DUMPALLMAPSXXX\_PROC**](#gt_a85344db-a5da-41e9-a60c-36d344a282b5) enumeration request, the values returned to the client MUST be different from the values returned for the previous request to GETCURRENTVERSIONTOKEN\_PROC or DUMPALLMAPSXXX\_PROC. (The method used to track changes in account mappings is implementation specific.) If the account mappings have not changed, the values returned to the client MUST be the values returned for the previous request to GETCURRENTVERSIONTOKEN\_PROC or DUMPALLMAPSXXX\_PROC.[<14>](#Appendix_A_14" \o "Product behavior note 14)

### Timer Events

None.

### Other Local Events

None.

# Protocol Examples

Several examples of network traffic for common User Name Mapping Protocol [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) procedures are outlined in the following sections, giving an indication of normal traffic flow. The example network traffic is illustrated with the aid of the following sample user and group mapping database at the server. In this example, a sample User Name Mapping Protocol SUNRPC service has been configured on the server to map users in the Windows domain "nfs-dom-1" to [**POSIX**](#gt_b1c15a13-c6a7-40bb-a0db-1947a755118d) user and [**group identifiers**](#gt_323122fc-2cf0-42be-98c6-cb492663c44b).

Advanced [**User Mappings**](#gt_6f6cefa4-0c32-4524-a267-2ce2cb29d7e9)

| Windows user | POSIX user | UID | GID |
| --- | --- | --- | --- |
| nfs-dom-1\administrator | Root | 0 | 0 |
| nfs-dom-1\u1 | u1 | 401 | 401 |
| nfs-dom-1\u2 | u2 | 402 | 401 |
| nfs-dom-1\u3 | u3 | 403 | 402 |

Advanced [**Group Mappings**](#gt_5501e0d8-1bdc-464c-8c40-c78de7b96734)

| Windows group | POSIX group | GID |
| --- | --- | --- |
| nfs-dom-1\Domain Admins | bin | 1 |
| nfs-dom-1\g1 | g1 | 401 |
| nfs-dom-1\g2 | g3 | 402 |

Simple User Mappings

| Windows user | POSIX user | UID | GID |
| --- | --- | --- | --- |
| nfs-dom-1\spec | spec | 500 | 500 |
| nfs-dom-1\u4 | u4 | 404 | 402 |
| nfs-dom-1\u5 | u5 | 405 | 401 |
| nfs-dom-1\u6 | u6 | 406 | 402 |

Simple Group Mappings

| Windows group | POSIX group | GID |
| --- | --- | --- |
| nfs-dom-1\specgroup | specgroup | 500 |
| nfs-dom-1\g4 | g4 | 404 |

## GETWINDOWSCREDSFROMUNIXUSERNAME\_PROC

The client sends a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) packet to the User Name Mapping Protocol service requesting the Windows account mapping for [**POSIX**](#gt_b1c15a13-c6a7-40bb-a0db-1947a755118d) user "root". The client asks for a match on the POSIX username alone in the **SearchOption** of the procedure.

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 33455, Total IP Length = 88
4. + Udp: SrcPort = 965, DstPort = UNM(819), Length = 68
5. - Rpc: Call, Program = mapsvc, Procedure =
6. GETWINDOWSCREDSFROMUNIXUSERNAME\_PROC
7. TransactionID: 1221413202 (0x48CD4952)
8. MessageType: Call
9. - ServiceCall:
10. RPCVersionNumber: 2 (0x2)
11. ProgramNumber: mapsvc, 351455(0x00055CDF)
12. ProgramVersion: 2 (0x2)
13. ProcedureNumber: GETWINDOWSCREDSFROMUNIXUSERNAME\_PROC
14. - Credential: No Identity Authentication
15. Flavor: No Identity Authentication
16. AuthDataLength: 0 (0x0)
17. - Verification:
18. Flavor: No Identity Authentication
19. AuthDataLength: 0 (0x0)
20. - Unm: GETWINDOWSCREDSFROMUNIXUSERNAME\_PROC Call
21. - UnixUser:
22. SearchOption: UnixAccountName is valid (0x1)
23. Reserved: Send as 0x00000000
24. ID: 0
25. - UnixAccountName: 0x1
26. - UNMName: root
27. Length: 4
28. Data: root

The User Name Mapping Protocol service on the server responds with a SUNRPC response packet with the [**advanced map**](#gt_2ee13d81-4d2a-44a0-8df0-aa38bc9f556d) for POSIX user "root" to Windows user "nfs-dom-1\administrator", illustrated as follows.

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 62340, Total IP Length = 88
4. + Udp: SrcPort = UNM(819), DstPort = 965, Length = 68
5. - Rpc: Reply, Status = Message accepted, Detail = Call succeeded
6. TransactionID: 1221413202 (0x48CD4952)
7. MessageType: Reply
8. - ServiceReply:
9. ReplyStatus: Message accepted
10. - MessageAccepted:
11. - Verification:
12. Flavor: No Identity Authentication
13. AuthDataLength: 0 (0x0)
14. AcceptState: Call succeeded
15. - Unm: GETWINDOWSCREDSFROMUNIXUSERNAME\_PROC Reply
16. - WindowsCreds:
17. Status: 0
18. Reserved: Send as 0x00000000
19. - WindowsAccountName: 0x1
20. - UNMWindowsName: nfs-dom-1\administrator
21. Length: 23
22. Data: nfs-dom-1\administrator
23. Padding: Binary Large Object (1 Bytes)

## GETUNIXCREDSFROMNTUSERNAME\_PROC

The client sends a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) packet to the User Name Mapping Protocol service that requests the [**POSIX**](#gt_b1c15a13-c6a7-40bb-a0db-1947a755118d) account mapping for the Windows user "nfs-dom-1\administrator".

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 40198, Total IP Length = 96
4. + Udp: SrcPort = 965, DstPort = UNM(819), Length = 76
5. - Rpc: Call, Program = mapsvc, Procedure =
6. GETUNIXCREDSFROMNTUSERNAME\_PROC
7. TransactionID: 1305299282 (0x4DCD4952)
8. MessageType: Call
9. - ServiceCall:
10. RPCVersionNumber: 2 (0x2)
11. ProgramNumber: mapsvc, 351455(0x00055CDF)
12. ProgramVersion: 2 (0x2)
13. ProcedureNumber: GETUNIXCREDSFROMNTUSERNAME\_PROC
14. - Credential: No Identity Authentication
15. Flavor: No Identity Authentication
16. AuthDataLength: 0 (0x0)
17. - Verification:
18. Flavor: No Identity Authentication
19. AuthDataLength: 0 (0x0)
20. - Unm: GETUNIXCREDSFROMNTUSERNAME\_PROC Call
21. - WindowsUserAccountName:
22. - WindowsAccountName: 0x1
23. - UNMName: nfs-dom-1\administrator
24. Length: 23
25. Data: nfs-dom-1\administrator
26. Padding: Binary Large Object (1 Bytes)

The User Name Mapping Protocol service on the server responds with a SUNRPC response packet with the [**advanced map**](#gt_2ee13d81-4d2a-44a0-8df0-aa38bc9f556d) for Windows user "nfs-dom-1\administrator" to POSIX user "root" with UID 0 and [**GID**](#gt_323122fc-2cf0-42be-98c6-cb492663c44b) 0, illustrated as follows.

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 20813, Total IP Length = 76
4. + Udp: SrcPort = UNM(819), DstPort = 965, Length = 56
5. - Rpc: Reply, Status = Message accepted, Detail = Call succeeded
6. TransactionID: 1305299282 (0x4DCD4952)
7. MessageType: Reply
8. - ServiceReply:
9. ReplyStatus: Message accepted
10. - MessageAccepted:
11. - Verification:
12. Flavor: No Identity Authentication
13. AuthDataLength: 0 (0x0)
14. AcceptState: Call succeeded
15. - Unm: GETUNIXCREDSFROMNTUSERNAME\_PROC Reply
16. - UnixCreds:
17. - UnixAccountName: 0x1
18. - UNMName: root
19. Length: 4
20. Data: root
21. ID: 0
22. GidCount: 2
23. - GID:
24. GID: 1
25. GID: 1

## AUTHUSINGUNIXCREDS\_PROC

The client sends a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) packet to the User Name Mapping Protocol service requesting the [**POSIX**](#gt_b1c15a13-c6a7-40bb-a0db-1947a755118d) account details for the POSIX user "root" with an empty password.

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 41135, Total IP Length = 80
4. + Udp: SrcPort = 965, DstPort = UNM(819), Length = 60
5. - Rpc: Call, Program = mapsvc, Procedure = AUTHUSINGUNIXCREDS\_PROC
6. TransactionID: 1322076498 (0x4ECD4952)
7. MessageType: Call
8. - ServiceCall:
9. RPCVersionNumber: 2 (0x2)
10. ProgramNumber: mapsvc, 351455(0x00055CDF)
11. ProgramVersion: 2 (0x2)
12. ProcedureNumber: AUTHUSINGUNIXCREDS\_PROC
13. - Credential: No Identity Authentication
14. Flavor: No Identity Authentication
15. AuthDataLength: 0 (0x0)
16. - Verification:
17. Flavor: No Identity Authentication
18. AuthDataLength: 0 (0x0)
19. - Unm: AUTHUSINGUNIXCREDS\_PROC Call
20. - UnixUserAuth:
21. - UnixUserAccountName: 0x1
22. - UNMName: root
23. Length: 4
24. Data: root
25. - UnixUserAccountPassword: 0x1
26. - UNMName:
27. Length: 0
28. Data:

The User Name Mapping Protocol service on the server responds with a SUNRPC response packet with the mapped POSIX account details for the user "root", illustrated as follows.

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 25985, Total IP Length = 76
4. + Udp: SrcPort = UNM(819), DstPort = 965, Length = 56
5. - Rpc: Reply, Status = Message accepted, Detail = Call succeeded
6. TransactionID: 1322076498 (0x4ECD4952)
7. MessageType: Reply
8. - ServiceReply:
9. ReplyStatus: Message accepted
10. - MessageAccepted:
11. - Verification:
12. Flavor: No Identity Authentication
13. AuthDataLength: 0 (0x0)
14. AcceptState: Call succeeded
15. - Unm: AUTHUSINGUNIXCREDS\_PROC Reply
16. - UnixCreds:
17. - UnixUserAccountPassword: 0x1
18. - UNMName: x
19. Length: 1
20. Data: x
21. Padding: Binary Large Object (3 Bytes)
22. ID: 0
23. GidCount: 2
24. - GID:
25. GID: 1
26. GID: 1

## DUMPALLMAPS\_PROC

The client sends a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) packet to the User Name Mapping Protocol service that requests an enumeration of all user maps (**PrincipalType**=0) starting at index zero (**MapRecordIndex**=0).

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 57181, Total IP Length = 76
4. + Udp: SrcPort = 965, DstPort = UNM(819), Length = 56
5. - Rpc: Call, Program = mapsvc, Procedure = DUMPALLMAPS\_PROC
6. TransactionID: 1238190418 (0x49CD4952)
7. MessageType: Call
8. - ServiceCall:
9. RPCVersionNumber: 2 (0x2)
10. ProgramNumber: mapsvc, 351455(0x00055CDF)
11. ProgramVersion: 2 (0x2)
12. ProcedureNumber: DUMPALLMAPS\_PROC
13. - Credential: No Identity Authentication
14. Flavor: No Identity Authentication
15. AuthDataLength: 0 (0x0)
16. - Verification:
17. Flavor: No Identity Authentication
18. AuthDataLength: 0 (0x0)
19. - Unm: DUMPALLMAPS\_PROC Call
20. - EnumCursor:
21. PrincipalType: Enumerate user account mappings (0)
22. MapRecordIndex: 0

The User Name Mapping Protocol service on the server responds with a listing of advanced and simple user mappings in the database. The response packet includes a sequence number that indicates the version for the current set of account mappings, a record count that indicates the number of mappings returned as a part of the current packet payload, the total number of maps in the database of the requested types, and finally, the individual maps themselves.

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 45847, Total IP Length = 308
4. + Udp: SrcPort = UNM(819), DstPort = 965, Length = 288
5. - Rpc: Reply, Status = Message accepted, Detail = Call succeeded
6. TransactionID: 1238190418 (0x49CD4952)
7. MessageType: Reply
8. - ServiceReply:
9. ReplyStatus: Message accepted
10. - MessageAccepted:
11. - Verification:
12. Flavor: No Identity Authentication
13. AuthDataLength: 0 (0x0)
14. AcceptState: Call succeeded
15. - Unm: DUMPALLMAPS\_PROC Reply
16. - Mapping:
17. - Token:
18. CurrentVersionTokenLowPart: 19924186
19. CurrentVersionTokenHighPart: 0
20. MappingRecordCount: 8
21. TotalMappingRecordCount: 8
22. - Map:
23. - WindowsAccountName: 0x1
24. - UNMName: nfs-dom-1\administrator
25. Length: 23
26. Data: nfs-dom-1\administrator
27. Padding: Binary Large Object (1 Bytes)
28. - UnixAccountName: 0x1
29. - UNMName: root
30. Length: 4
31. Data: root
32. ID: 0
33. - Map:
34. - WindowsAccountName: 0x1
35. - UNMName: NFS-DOM-1\u1
36. Length: 12
37. Data: NFS-DOM-1\u1
38. - UnixAccountName: 0x1
39. - UNMName: u1
40. Length: 2
41. Data: u1
42. Padding: Binary Large Object (2 Bytes)
43. ID: 401
44. - Map:
45. - WindowsAccountName: 0x1
46. - UNMName: NFS-DOM-1\u2
47. Length: 12
48. Data: NFS-DOM-1\u2
49. - UnixAccountName: 0x1
50. - UNMName: u2
51. Length: 2
52. Data: u2
53. Padding: Binary Large Object (2 Bytes)
54. ID: 402
55. - Map:
56. - WindowsAccountName: 0x1
57. - UNMName: NFS-DOM-1\u3
58. Length: 12
59. Data: NFS-DOM-1\u3
60. - UnixAccountName: 0x1
61. - UNMName: u3
62. Length: 2
63. Data: u3
64. Padding: Binary Large Object (2 Bytes)
65. ID: 403
66. - Map:
67. - WindowsAccountName: 0x1
68. - UNMName: NFS-DOM-1\spec
69. Length: 14
70. Data: NFS-DOM-1\spec
71. Padding: Binary Large Object (2 Bytes)
72. - UnixAccountName: 0x1
73. - UNMName: spec
74. Length: 4
75. Data: spec
76. ID: 500
77. - Map:
78. - WindowsAccountName: 0x1
79. - UNMName: NFS-DOM-1\u4
80. Length: 12
81. Data: NFS-DOM-1\u4
82. - UnixAccountName: 0x1
83. - UNMName: u4
84. Length: 2
85. Data: u4
86. Padding: Binary Large Object (2 Bytes)
87. ID: 404
88. - Map:
89. - WindowsAccountName: 0x1
90. - UNMName: NFS-DOM-1\u5
91. Length: 12
92. Data: NFS-DOM-1\u5
93. - UnixAccountName: 0x1
94. - UNMName: u5
95. Length: 2
96. Data: u5
97. Padding: Binary Large Object (2 Bytes)
98. ID: 405
99. - Map:
100. - WindowsAccountName: 0x1
101. - UNMName: NFS-DOM-1\u6
102. Length: 12
103. Data: NFS-DOM-1\u6
104. - UnixAccountName: 0x1
105. - UNMName: u6
106. Length: 2
107. Data: u6
108. Padding: Binary Large Object (2 Bytes)
109. ID: 406

## GETCURRENTVERSIONTOKEN\_PROC

The client sends a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) packet to the User Name Mapping Protocol service that requests the current account mapping sequence number for the set of mapping records held by the server.

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 47908, Total IP Length = 76
4. + Udp: SrcPort = 965, DstPort = UNM(819), Length = 56
5. - Rpc: Call, Program = mapsvc, Procedure = GETCURRENTVERSIONTOKEN\_PROC
6. TransactionID: 1422739794 (0x54CD4952)
7. MessageType: Call
8. - ServiceCall:
9. RPCVersionNumber: 2 (0x2)
10. ProgramNumber: mapsvc, 351455(0x00055CDF)
11. ProgramVersion: 2 (0x2)
12. ProcedureNumber: GETCURRENTVERSIONTOKEN\_PROC
13. - Credential: No Identity Authentication
14. Flavor: No Identity Authentication
15. AuthDataLength: 0 (0x0)
16. - Verification:
17. Flavor: No Identity Authentication
18. AuthDataLength: 0 (0x0)
19. - Unm: GETCURRENTVERSIONTOKEN\_PROC Call
20. - SequenceNumber:
21. CurrentVersionTokenLowPart: 11337900
22. CurrentVersionTokenHighPart: 0

The User Name Mapping Protocol service on the server responds with a SUNRPC response packet with the current sequence number value for the set of mapping records held by it, illustrated as follows.

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 38726, Total IP Length = 60
4. + Udp: SrcPort = UNM(819), DstPort = 965, Length = 40
5. - Rpc: Reply, Status = Message accepted, Detail = Call succeeded
6. TransactionID: 1422739794 (0x54CD4952)
7. MessageType: Reply
8. - ServiceReply:
9. ReplyStatus: Message accepted
10. - MessageAccepted:
11. - Verification:
12. Flavor: No Identity Authentication
13. AuthDataLength: 0 (0x0)
14. AcceptState: Call succeeded
15. - Unm: GETCURRENTVERSIONTOKEN\_PROC Reply
16. - SequenceNumber:
17. CurrentVersionTokenLowPart: 19924186
18. CurrentVersionTokenHighPart: 0

## DUMPALLMAPSEX\_PROC

The client sends a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) packet to the User Name Mapping Protocol service requesting an enumeration of all user maps (**PrincipalType**=0) starting at index zero (**MapRecordIndex**=0).

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 48740, Total IP Length = 76
4. + Udp: SrcPort = 965, DstPort = UNM(819), Length = 56
5. - Rpc: Call, Program = mapsvc, Procedure = DUMPALLMAPSEX\_PROC
6. TransactionID: 1439517010 (0x55CD4952)
7. MessageType: Call
8. - ServiceCall:
9. RPCVersionNumber: 2 (0x2)
10. ProgramNumber: mapsvc, 351455(0x00055CDF)
11. ProgramVersion: 2 (0x2)
12. ProcedureNumber: DUMPALLMAPSEX\_PROC
13. - Credential: No Identity Authentication
14. Flavor: No Identity Authentication
15. AuthDataLength: 0 (0x0)
16. - Verification:
17. Flavor: No Identity Authentication
18. AuthDataLength: 0 (0x0)
19. - Unm: DUMPALLMAPSEX\_PROC Call
20. - EnumCursor:
21. PrincipalType: Enumerate user account mappings
22. MapRecordIndex: 0

The User Name Mapping Protocol service on the server responds with a listing of advanced and simple user mappings in the database. The response packet includes a sequence number that indicates the version for the current set of account mappings, a record count that indicates the number of mappings returned as a part of the current packet payload, the total number of maps in the database of the requested types, and finally, the individual maps themselves.

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 42795, Total IP Length = 464
4. + Udp: SrcPort = UNM(819), DstPort = 965, Length = 444
5. - Rpc: Reply, Status = Message accepted, Detail = Call succeeded
6. TransactionID: 1439517010 (0x55CD4952)
7. MessageType: Reply
8. - ServiceReply:
9. ReplyStatus: Message accepted
10. - MessageAccepted:
11. - Verification:
12. Flavor: No Identity Authentication
13. AuthDataLength: 0 (0x0)
14. AcceptState: Call succeeded
15. - Unm: DUMPALLMAPSEX\_PROC Reply
16. - Maps: Count = 8
17. - Token:
18. CurrentVersionTokenLowPart: 19924186
19. CurrentVersionTokenHighPart: 0
20. MappingRecordCount: 8
21. TotalMappingRecordCount: 8
22. - Map: 0x1
23. - UNMMapString: \*:nfs-dom-1\administrator:0:PCNFS:PCNFS:
24. root:x:0:1:1
25. Length: 52
26. Data: \*:nfs-dom-1\administrator:0:PCNFS:PCNFS:root:x:0:1:1
27. - Map: 0x1
28. - UNMMapString: \*:NFS-DOM-1\u1:0:PCNFS:PCNFS:u1:x:401:401
29. Length: 41
30. Data: \*:NFS-DOM-1\u1:0:PCNFS:PCNFS:u1:x:401:401
31. Padding: Binary Large Object (3 Bytes)
32. - Map: 0x1
33. - UNMMapString: \*:NFS-DOM-1\u2:0:PCNFS:PCNFS:u2:x:402:401
34. Length: 41
35. Data: \*:NFS-DOM-1\u2:0:PCNFS:PCNFS:u2:x:402:401
36. Padding: Binary Large Object (3 Bytes)
37. - Map: 0x1
38. - UNMMapString: \*:NFS-DOM-1\u3:0:PCNFS:PCNFS:u3:x:403:402
39. Length: 41
40. Data: \*:NFS-DOM-1\u3:0:PCNFS:PCNFS:u3:x:403:402
41. Padding: Binary Large Object (3 Bytes)
42. - Map: 0x1
43. - UNMMapString: -:NFS-DOM-1\spec:0:PCNFS:PCNFS:spec:x:500:500
44. Length: 45
45. Data: -:NFS-DOM-1\spec:0:PCNFS:PCNFS:spec:x:500:500
46. Padding: Binary Large Object (3 Bytes)
47. - Map: 0x1
48. - UNMMapString: -:NFS-DOM-1\u4:0:PCNFS:PCNFS:u4:x:404:402
49. Length: 41
50. Data: -:NFS-DOM-1\u4:0:PCNFS:PCNFS:u4:x:404:402
51. Padding: Binary Large Object (3 Bytes)
52. - Map: 0x1
53. - UNMMapString: -:NFS-DOM-1\u5:0:PCNFS:PCNFS:u5:x:405:401
54. Length: 41
55. Data: -:NFS-DOM-1\u5:0:PCNFS:PCNFS:u5:x:405:401
56. Padding: Binary Large Object (3 Bytes)
57. - Map: 0x1
58. - UNMMapString: -:NFS-DOM-1\u6:0:PCNFS:PCNFS:u6:x:406:402
59. Length: 41
60. Data: -:NFS-DOM-1\u6:0:PCNFS:PCNFS:u6:x:406:402
61. Padding: Binary Large Object (3 Bytes)

## GETWINDOWSGROUPFROMUNIXGROUPNAME\_PROC

The client sends a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) packet to the User Name Mapping Protocol service requesting the Windows group mapping for [**POSIX**](#gt_b1c15a13-c6a7-40bb-a0db-1947a755118d) group "bin".

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 53170, Total IP Length = 88
4. + Udp: SrcPort = 965, DstPort = UNM(819), Length = 68
5. - Rpc: Call, Program = mapsvc, Procedure =
6. GETNTCREDSFROMUNIXGROUPNAME\_PROC
7. TransactionID: 1473071442 (0x57CD4952)
8. MessageType: Call
9. - ServiceCall:
10. RPCVersionNumber: 2 (0x2)
11. ProgramNumber: mapsvc, 351455(0x00055CDF)
12. ProgramVersion: 2 (0x2)
13. ProcedureNumber: GETNTCREDSFROMUNIXGROUPNAME\_PROC
14. - Credential: No Identity Authentication
15. Flavor: No Identity Authentication
16. AuthDataLength: 0 (0x0)
17. - Verification:
18. Flavor: No Identity Authentication
19. AuthDataLength: 0 (0x0)
20. - Unm: GETNTCREDSFROMUNIXGROUPNAME\_PROC Call
21. - UnixGroupAccount:
22. SearchOption: UnixAccountName and ID are both valid
23. Reserved: Send as 0x00000000
24. ID: 1
25. - UnixAccountName: 0x1
26. - UNMName: bin
27. Length: 3
28. Data: bin
29. Padding: Binary Large Object (1 Bytes)

The User Name Mapping Protocol service on the server responds with a SUNRPC response packet with the [**group map**](#gt_5501e0d8-1bdc-464c-8c40-c78de7b96734) for POSIX group "bin" to Windows group "nfs-dom-1\Domain Admins", illustrated as follows.

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 49784, Total IP Length = 88
4. + Udp: SrcPort = UNM(819), DstPort = 965, Length = 68
5. - Rpc: Reply, Status = Message accepted, Detail = Call succeeded
6. TransactionID: 1473071442 (0x57CD4952)
7. MessageType: Reply
8. - ServiceReply:
9. ReplyStatus: Message accepted
10. - MessageAccepted:
11. - Verification:
12. Flavor: No Identity Authentication
13. AuthDataLength: 0 (0x0)
14. AcceptState: Call succeeded
15. - Unm: GETNTCREDSFROMUNIXGROUPNAME\_PROC Reply
16. - WindowsCreds:
17. Status: 0
18. Reserved: Send as 0x00000000
19. - WindowsAccountName: 0x1
20. - UNMWindowsName: NFS-DOM-1\Domain Admins
21. Length: 23
22. Data: NFS-DOM-1\Domain Admins
23. Padding: Binary Large Object (1 Bytes)

## GETUNIXCREDSFROMNTGROUPNAME\_PROC

The client sends a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) packet to the User Name Mapping Protocol service requesting the [**POSIX**](#gt_b1c15a13-c6a7-40bb-a0db-1947a755118d) group mapping for the Windows group "nfs-dom-1\g1".

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 54821, Total IP Length = 84
4. + Udp: SrcPort = 965, DstPort = UNM(819), Length = 64
5. - Rpc: Call, Program = mapsvc, Procedure =
6. GETUNIXCREDSFROMNTGROUPNAME\_PROC
7. TransactionID: 1489848658 (0x58CD4952)
8. MessageType: Call
9. - ServiceCall:
10. RPCVersionNumber: 2 (0x2)
11. ProgramNumber: mapsvc, 351455(0x00055CDF)
12. ProgramVersion: 2 (0x2)
13. ProcedureNumber: GETUNIXCREDSFROMNTGROUPNAME\_PROC
14. - Credential: No Identity Authentication
15. Flavor: No Identity Authentication
16. AuthDataLength: 0 (0x0)
17. - Verification:
18. Flavor: No Identity Authentication
19. AuthDataLength: 0 (0x0)
20. - Unm: GETUNIXCREDSFROMNTGROUPNAME\_PROC Call
21. - WindowsGroupAccountName:
22. - WindowsAccountName: 0x1
23. - UNMName: nfs-dom-1\g1
24. Length: 12
25. Data: nfs-dom-1\g1

The User Name Mapping Protocol service on the server responds with a SUNRPC response packet with the [**group map**](#gt_5501e0d8-1bdc-464c-8c40-c78de7b96734) for Windows group "nfs-dom-1\g1" to the POSIX group "g1", illustrated as follows.

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 50256, Total IP Length = 68
4. + Udp: SrcPort = UNM(819), DstPort = 965, Length = 48
5. - Rpc: Reply, Status = Message accepted, Detail = Call succeeded
6. TransactionID: 1489848658 (0x58CD4952)
7. MessageType: Reply
8. - ServiceReply:
9. ReplyStatus: Message accepted
10. - MessageAccepted:
11. - Verification:
12. Flavor: No Identity Authentication
13. AuthDataLength: 0 (0x0)
14. AcceptState: Call succeeded
15. - Unm: GETUNIXCREDSFROMNTGROUPNAME\_PROC Reply
16. - UnixCreds:
17. - UnixAccountName: 0x1
18. - UNMName: g1
19. Length: 2
20. Data: g1
21. Padding: Binary Large Object (2 Bytes)
22. ID: 401
23. GidCount: 0

## GETUNIXCREDSFROMNTUSERSID\_PROC

The client sends a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) packet to the User Name Mapping Protocol service requesting the [**POSIX**](#gt_b1c15a13-c6a7-40bb-a0db-1947a755118d) credentials for the Windows user [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) "S-1-5-21-3994172400-2625080034-4079281819-500" that represents Windows user account "nfs-dom-1\administrator".

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 51864, Total IP Length = 100
4. + Udp: SrcPort = 1013, DstPort = UNM(819), Length = 80
5. - Rpc: Call, Program = mapsvc, Procedure =
6. GETUNIXCREDSFROMNTUSERSID\_PROC
7. TransactionID: 1238234037 (0x49CDF3B5)
8. MessageType: Call
9. - ServiceCall:
10. RPCVersionNumber: 2 (0x2)
11. ProgramNumber: mapsvc, 351455(0x00055CDF)
12. ProgramVersion: 2 (0x2)
13. ProcedureNumber: GETUNIXCREDSFROMNTUSERSID\_PROC
14. - Credential: No Identity Authentication
15. Flavor: No Identity Authentication
16. AuthDataLength: 0 (0x0)
17. - Verification:
18. Flavor: No Identity Authentication
19. AuthDataLength: 0 (0x0)
20. - Unm: GETUNIXCREDSFROMNTUSERSID\_PROC Call
21. - Sid:
22. Sidlength: 28
23. SID: 01 05 00 00 00 00 00 05 15 00 00 00 F0 3B 12 EE
24. E2 8A 77 9C 9B E6 24 F3 F4 01 00 00

The User Name Mapping Protocol service on the server responds with a SUNRPC response packet with the POSIX credentials for the mapped [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account that corresponds to Windows user "nfs-dom-1\Administrator" as POSIX user "root", illustrated as follows.

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 14698, Total IP Length = 76
4. + Udp: SrcPort = UNM(819), DstPort = 1013, Length = 56
5. - Rpc: Reply, Status = Message accepted, Detail = Call succeeded
6. TransactionID: 1238234037 (0x49CDF3B5)
7. MessageType: Reply
8. - ServiceReply:
9. ReplyStatus: Message accepted
10. - MessageAccepted:
11. - Verification:
12. Flavor: No Identity Authentication
13. AuthDataLength: 0 (0x0)
14. AcceptState: Call succeeded
15. - Unm: GETUNIXCREDSFROMNTUSERSID\_PROC Reply
16. - UnixCreds:
17. - UnixAccountName: 0x1
18. - UNMName: root
19. Length: 4
20. Data: root
21. ID: 0
22. GidCount: 2
23. - GID:
24. GID: 1
25. GID: 1

## DUMPALLMAPSW\_PROC

The client sends a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) packet to the User Name Mapping Protocol service requesting an enumeration of all user maps (**PrincipalType**=0) starting at index zero (**MapRecordIndex**=0).

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 59252, Total IP Length = 76
4. + Udp: SrcPort = 965, DstPort = UNM(819), Length = 56
5. - Rpc: Call, Program = mapsvc, Procedure = DUMPALLMAPSW\_PROC
6. TransactionID: 1590511954 (0x5ECD4952)
7. MessageType: Call
8. - ServiceCall:
9. RPCVersionNumber: 2 (0x2)
10. ProgramNumber: mapsvc, 351455(0x00055CDF)
11. ProgramVersion: 2 (0x2)
12. ProcedureNumber: DUMPALLMAPSW\_PROC
13. - Credential: No Identity Authentication
14. Flavor: No Identity Authentication
15. AuthDataLength: 0 (0x0)
16. - Verification:
17. Flavor: No Identity Authentication
18. AuthDataLength: 0 (0x0)
19. - Unm: DUMPALLMAPSW\_PROC Call
20. - EnumCursor:
21. PrincipalType: Enumerate user account mappings
22. MapRecordIndex: 0

The User Name Mapping Protocol service on the server responds with a listing of advanced and simple user mappings in the database. The response packet includes a sequence number that indicates the version for the current set of account mappings, a record count that indicates the number of mappings returned as a part of the current packet payload, the total number of maps in the database of the requested types, and finally, the individual maps themselves.

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 55477, Total IP Length = 424
4. + Udp: SrcPort = UNM(819), DstPort = 965, Length = 404
5. - Rpc: Reply, Status = Message accepted, Detail = Call succeeded
6. TransactionID: 1590511954 (0x5ECD4952)
7. MessageType: Reply
8. - ServiceReply:
9. ReplyStatus: Message accepted
10. - MessageAccepted:
11. - Verification:
12. Flavor: No Identity Authentication
13. AuthDataLength: 0 (0x0)
14. AcceptState: Call succeeded
15. - Unm: DUMPALLMAPSW\_PROC Reply
16. - MappingW:
17. - Token:
18. CurrentVersionTokenLowPart: 19924186
19. CurrentVersionTokenHighPart: 0
20. MappingRecordCount: 8
21. TotalMappingRecordCount: 8
22. - Map:
23. - WindowsAccountName: 0x1
24. - UNMNameW: nfs-dom-1\administrator
25. Length: 46
26. Data: nfs-dom-1\administrator
27. Padding: Binary Large Object (2 Bytes)
28. - UnixAccountName: 0x1
29. - UNMNameW: root
30. Length: 8
31. Data: root
32. ID: 0
33. - Map:
34. - WindowsAccountName: 0x1
35. - UNMNameW: NFS-DOM-1\u1
36. Length: 24
37. Data: NFS-DOM-1\u1
38. - UnixAccountName: 0x1
39. - UNMNameW: u1
40. Length: 4
41. Data: u1
42. ID: 401
43. - Map:
44. - WindowsAccountName: 0x1
45. - UNMNameW: NFS-DOM-1\u2
46. Length: 24
47. Data: NFS-DOM-1\u2
48. - UnixAccountName: 0x1
49. - UNMNameW: u2
50. Length: 4
51. Data: u2
52. ID: 402
53. - Map:
54. - WindowsAccountName: 0x1
55. - UNMNameW: NFS-DOM-1\u3
56. Length: 24
57. Data: NFS-DOM-1\u3
58. - UnixAccountName: 0x1
59. - UNMNameW: u3
60. Length: 4
61. Data: u3
62. ID: 403
63. - Map:
64. - WindowsAccountName: 0x1
65. - UNMNameW: NFS-DOM-1\spec
66. Length: 28
67. Data: NFS-DOM-1\spec
68. - UnixAccountName: 0x1
69. - UNMNameW: spec
70. Length: 8
71. Data: spec
72. ID: 500
73. - Map:
74. - WindowsAccountName: 0x1
75. - UNMNameW: NFS-DOM-1\u4
76. Length: 24
77. Data: NFS-DOM-1\u4
78. - UnixAccountName: 0x1
79. - UNMNameW: u4
80. Length: 4
81. Data: u4
82. ID: 404
83. - Map:
84. - WindowsAccountName: 0x1
85. - UNMNameW: NFS-DOM-1\u5
86. Length: 24
87. Data: NFS-DOM-1\u5
88. - UnixAccountName: 0x1
89. - UNMNameW: u5
90. Length: 4
91. Data: u5
92. ID: 405
93. - Map:
94. - WindowsAccountName: 0x1
95. - UNMNameW: NFS-DOM-1\u6
96. Length: 24
97. Data: NFS-DOM-1\u6
98. - UnixAccountName: 0x1
99. - UNMNameW: u6
100. Length: 4
101. Data: u6
102. ID: 406

## DUMPALLMAPSEXW\_PROC

The client sends a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) packet to the User Name Mapping Protocol service requesting an enumeration of all user maps (**PrincipalType**=0) starting at index zero (**MapRecordIndex**=0).

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 60653, Total IP Length = 76
4. + Udp: SrcPort = 965, DstPort = UNM(819), Length = 56
5. - Rpc: Call, Program = mapsvc, Procedure = DUMPALLMAPSEXW\_PROC
6. TransactionID: 1607289170 (0x5FCD4952)
7. MessageType: Call
8. - ServiceCall:
9. RPCVersionNumber: 2 (0x2)
10. ProgramNumber: mapsvc, 351455(0x00055CDF)
11. ProgramVersion: 2 (0x2)
12. ProcedureNumber: DUMPALLMAPSEXW\_PROC
13. - Credential: No Identity Authentication
14. Flavor: No Identity Authentication
15. AuthDataLength: 0 (0x0)
16. - Verification:
17. Flavor: No Identity Authentication
18. AuthDataLength: 0 (0x0)
19. - Unm: DUMPALLMAPSEXW\_PROC Call
20. - EnumCursor:
21. PrincipalType: Enumerate user account mappings
22. MapRecordIndex: 0

The User Name Mapping Protocol service on the server responds with a listing of advanced and simple user mappings in the database. The response packet includes a sequence number that indicates the version for the current set of account mappings, a record count that indicates the number of mappings returned as a part of the current packet payload, the total number of maps in the database of the requested types, and finally, the individual maps themselves.

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 56145, Total IP Length = 800
4. + Udp: SrcPort = UNM(819), DstPort = 965, Length = 780
5. - Rpc: Reply, Status = Message accepted, Detail = Call succeeded
6. TransactionID: 1607289170 (0x5FCD4952)
7. MessageType: Reply
8. - ServiceReply:
9. ReplyStatus: Message accepted
10. - MessageAccepted:
11. - Verification:
12. Flavor: No Identity Authentication
13. AuthDataLength: 0 (0x0)
14. AcceptState: Call succeeded
15. - Unm: DUMPALLMAPSEXW\_PROC Reply
16. - MapsW:
17. - Token:
18. CurrentVersionTokenLowPart: 19924186
19. CurrentVersionTokenHighPart: 0
20. MappingRecordCount: 8
21. TotalMappingRecordCount: 8
22. - Map: 0x1
23. - UNMMapStringW: \*:nfs-dom-1\administrator:0:PCNFS:PCNFS:
24. root:x:0:1:1
25. Length: 104
26. Data: \*:nfs-dom-1\administrator:0:PCNFS:PCNFS:root:x:0:1:1
27. - Map: 0x1
28. - UNMMapStringW: \*:NFS-DOM-1\u1:0:PCNFS:PCNFS:u1:x:401:401
29. Length: 82
30. Data: \*:NFS-DOM-1\u1:0:PCNFS:PCNFS:u1:x:401:401
31. Padding: Binary Large Object (2 Bytes)
32. - Map: 0x1
33. - UNMMapStringW: \*:NFS-DOM-1\u2:0:PCNFS:PCNFS:u2:x:402:401
34. Length: 82
35. Data: \*:NFS-DOM-1\u2:0:PCNFS:PCNFS:u2:x:402:401
36. Padding: Binary Large Object (2 Bytes)
37. - Map: 0x1
38. - UNMMapStringW: \*:NFS-DOM-1\u3:0:PCNFS:PCNFS:u3:x:403:402
39. Length: 82
40. Data: \*:NFS-DOM-1\u3:0:PCNFS:PCNFS:u3:x:403:402
41. Padding: Binary Large Object (2 Bytes)
42. - Map: 0x1
43. - UNMMapStringW: -:NFS-DOM-1\spec:0:PCNFS:PCNFS:spec:x:500:500
44. Length: 90
45. Data: -:NFS-DOM-1\spec:0:PCNFS:PCNFS:spec:x:500:500
46. Padding: Binary Large Object (2 Bytes)
47. - Map: 0x1
48. - UNMMapStringW: -:NFS-DOM-1\u4:0:PCNFS:PCNFS:u4:x:404:402
49. Length: 82
50. Data: -:NFS-DOM-1\u4:0:PCNFS:PCNFS:u4:x:404:402
51. Padding: Binary Large Object (2 Bytes)
52. - Map: 0x1
53. - UNMMapStringW: -:NFS-DOM-1\u5:0:PCNFS:PCNFS:u5:x:405:401
54. Length: 82
55. Data: -:NFS-DOM-1\u5:0:PCNFS:PCNFS:u5:x:405:401
56. Padding: Binary Large Object (2 Bytes)
57. - Map: 0x1
58. - UNMMapStringW: -:NFS-DOM-1\u6:0:PCNFS:PCNFS:u6:x:406:402
59. Length: 82
60. Data: -:NFS-DOM-1\u6:0:PCNFS:PCNFS:u6:x:406:402
61. Padding: Binary Large Object (2 Bytes)

## GETWINDOWSUSERFROMUNIXUSERNAMEW\_PROC

The client sends a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) packet to the User Name Mapping Protocol service requesting the Windows user mapping for [**POSIX**](#gt_b1c15a13-c6a7-40bb-a0db-1947a755118d) user "root".

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 61716, Total IP Length = 92
4. + Udp: SrcPort = 965, DstPort = UNM(819), Length = 72
5. - Rpc: Call, Program = mapsvc, Procedure =
6. GETNTCREDSFROMUNIXUSERNAMEW\_PROC
7. TransactionID: 1624066386 (0x60CD4952)
8. MessageType: Call
9. - ServiceCall:
10. RPCVersionNumber: 2 (0x2)
11. ProgramNumber: mapsvc, 351455(0x00055CDF)
12. ProgramVersion: 2 (0x2)
13. ProcedureNumber: GETNTCREDSFROMUNIXUSERNAMEW\_PROC
14. - Credential: No Identity Authentication
15. Flavor: No Identity Authentication
16. AuthDataLength: 0 (0x0)
17. - Verification:
18. Flavor: No Identity Authentication
19. AuthDataLength: 0 (0x0)
20. - Unm: GETNTCREDSFROMUNIXUSERNAMEW\_PROC Call
21. - UnixUserW:
22. SearchOption: UnixAccountName and ID are both valid
23. Reserved: Send as 0x00000000
24. ID: 0
25. - UnixAccountName: 0x1
26. - UNMNameW: root
27. Length: 8
28. Data: root

The User Name Mapping Protocol service on the server responds with a SUNRPC response packet with the [**user map**](#gt_6f6cefa4-0c32-4524-a267-2ce2cb29d7e9) for POSIX user "root" to Windows user "nfs-dom-1\Administrator", illustrated as follows.

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 60521, Total IP Length = 112
4. + Udp: SrcPort = UNM(819), DstPort = 965, Length = 92
5. - Rpc: Reply, Status = Message accepted, Detail = Call succeeded
6. TransactionID: 1624066386 (0x60CD4952)
7. MessageType: Reply
8. - ServiceReply:
9. ReplyStatus: Message accepted
10. - MessageAccepted:
11. - Verification:
12. Flavor: No Identity Authentication
13. AuthDataLength: 0 (0x0)
14. AcceptState: Call succeeded
15. - Unm: GETNTCREDSFROMUNIXUSERNAMEW\_PROC Reply
16. - WindowsCredsW:
17. Status: 0
18. Reserved: Send as 0x00000000
19. - WindowsAccountName: 0x1
20. - UNMWindowsNameW: nfs-dom-1\administrator
21. Length: 46
22. Data: nfs-dom-1\administrator
23. Padding: Binary Large Object (2 Bytes)

## GETUNIXCREDSFROMNTUSERNAMEW\_PROC

The client sends a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) packet to the User Name Mapping Protocol service requesting the [**POSIX**](#gt_b1c15a13-c6a7-40bb-a0db-1947a755118d) user mapping for the Windows user "nfs-dom-1\Administrator".

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 62611, Total IP Length = 120
4. + Udp: SrcPort = 965, DstPort = UNM(819), Length = 100
5. - Rpc: Call, Program = mapsvc, Procedure =
6. GETUNIXCREDSFROMNTUSERNAMEW\_PROC
7. TransactionID: 1640843602 (0x61CD4952)
8. MessageType: Call
9. - ServiceCall:
10. RPCVersionNumber: 2 (0x2)
11. ProgramNumber: mapsvc, 351455(0x00055CDF)
12. ProgramVersion: 2 (0x2)
13. ProcedureNumber: GETUNIXCREDSFROMNTUSERNAMEW\_PROC
14. - Credential: No Identity Authentication
15. Flavor: No Identity Authentication
16. AuthDataLength: 0 (0x0)
17. - Verification:
18. Flavor: No Identity Authentication
19. AuthDataLength: 0 (0x0)
20. - Unm: GETUNIXCREDSFROMNTUSERNAMEW\_PROC Call
21. - WindowsUserAccountNameW:
22. - WindowsAccountName: 0x1
23. - UNMNameW: nfs-dom-1\administrator
24. Length: 46
25. Data: nfs-dom-1\administrator
26. Padding: Binary Large Object (2 Bytes)

The User Name Mapping Protocol service on the server responds with a SUNRPC response packet with the [**user map**](#gt_6f6cefa4-0c32-4524-a267-2ce2cb29d7e9) for Windows user "nfs-dom-1\Administrator" to the POSIX user "root", illustrated as follows.

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 63086, Total IP Length = 80
4. + Udp: SrcPort = UNM(819), DstPort = 965, Length = 60
5. - Rpc: Reply, Status = Message accepted, Detail = Call succeeded
6. TransactionID: 1640843602 (0x61CD4952)
7. MessageType: Reply
8. - ServiceReply:
9. ReplyStatus: Message accepted
10. - MessageAccepted:
11. - Verification:
12. Flavor: No Identity Authentication
13. AuthDataLength: 0 (0x0)
14. AcceptState: Call succeeded
15. - Unm: GETUNIXCREDSFROMNTUSERNAMEW\_PROC Reply
16. - UnixCredsW:
17. - UnixAccountName: 0x1
18. - UNMNameW: root
19. Length: 8
20. Data: root
21. ID: 0
22. GidCount: 2
23. - GID:
24. GID: 1
25. GID: 1

## AUTHUSINGUNIXCREDSW\_PROC

The client sends a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) packet to the User Name Mapping Protocol service requesting the [**POSIX**](#gt_b1c15a13-c6a7-40bb-a0db-1947a755118d) account details for the POSIX user "root" with an empty password.

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 64478, Total IP Length = 84
4. + Udp: SrcPort = 965, DstPort = UNM(819), Length = 64
5. - Rpc: Call, Program = mapsvc, Procedure = AUTHUSINGUNIXCREDSW\_PROC
6. TransactionID: 1724729682 (0x66CD4952)
7. MessageType: Call
8. - ServiceCall:
9. RPCVersionNumber: 2 (0x2)
10. ProgramNumber: mapsvc, 351455(0x00055CDF)
11. ProgramVersion: 2 (0x2)
12. ProcedureNumber: AUTHUSINGUNIXCREDSW\_PROC
13. - Credential: No Identity Authentication
14. Flavor: No Identity Authentication
15. AuthDataLength: 0 (0x0)
16. - Verification:
17. Flavor: No Identity Authentication
18. AuthDataLength: 0 (0x0)
19. - Unm: AUTHUSINGUNIXCREDSW\_PROC Call
20. - UnixUserAuthW:
21. - UnixUserAccountName: 0x1
22. - UNMNameW: root
23. Length: 8
24. Data: root
25. - UnixUserAccountPassword: 0x1
26. - UNMNameW:
27. Length: 0
28. Data:

The User Name Mapping Protocol service on the server responds with a SUNRPC response packet with the mapped POSIX account details for the user "root", illustrated as follows.

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 5741, Total IP Length = 76
4. + Udp: SrcPort = UNM(819), DstPort = 965, Length = 56
5. - Rpc: Reply, Status = Message accepted, Detail = Call succeeded
6. TransactionID: 1724729682 (0x66CD4952)
7. MessageType: Reply
8. - ServiceReply:
9. ReplyStatus: Message accepted
10. - MessageAccepted:
11. - Verification:
12. Flavor: No Identity Authentication
13. AuthDataLength: 0 (0x0)
14. AcceptState: Call succeeded
15. - Unm: AUTHUSINGUNIXCREDSW\_PROC Reply
16. - UnixCredsW:
17. - UnixAccountPassword: 0x1
18. - UNMNameW: x
19. Length: 2
20. Data: x
21. Padding: Binary Large Object (2 Bytes)
22. ID: 0
23. GidCount: 2
24. - GID:
25. GID: 1
26. GID: 1

## GETWINDOWSGROUPFROMUNIXGROUPNAMEW\_PROC

The client sends a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) packet to the User Name Mapping Protocol service requesting the Windows group mapping for [**POSIX**](#gt_b1c15a13-c6a7-40bb-a0db-1947a755118d) group "g1".

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 65220, Total IP Length = 88
4. + Udp: SrcPort = 965, DstPort = UNM(819), Length = 68
5. - Rpc: Call, Program = mapsvc, Procedure =
6. GETNTCREDSFROMUNIXGROUPNAMEW\_PROC
7. TransactionID: 1741506898 (0x67CD4952)
8. MessageType: Call
9. - ServiceCall:
10. RPCVersionNumber: 2 (0x2)
11. ProgramNumber: mapsvc, 351455(0x00055CDF)
12. ProgramVersion: 2 (0x2)
13. ProcedureNumber: GETNTCREDSFROMUNIXGROUPNAMEW\_PROC
14. - Credential: No Identity Authentication
15. Flavor: No Identity Authentication
16. AuthDataLength: 0 (0x0)
17. - Verification:
18. Flavor: No Identity Authentication
19. AuthDataLength: 0 (0x0)
20. - Unm: GETNTCREDSFROMUNIXGROUPNAMEW\_PROC Call
21. - UnixGroupAccountW:
22. SearchOption: UnixAccountName and ID are both valid
23. Reserved: Send as 0x00000000
24. ID: 401
25. - UnixAccountName: 0x1
26. - UNMNameW: g1
27. Length: 4
28. Data: g1

The User Name Mapping Protocol service on the server responds with a SUNRPC response packet with the [**group map**](#gt_5501e0d8-1bdc-464c-8c40-c78de7b96734) for POSIX group "bin" to Windows group "nfs-dom-1\Domain Admins", illustrated as follows.

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 8813, Total IP Length = 88
4. + Udp: SrcPort = UNM(819), DstPort = 965, Length = 68
5. - Rpc: Reply, Status = Message accepted, Detail = Call succeeded
6. TransactionID: 1741506898 (0x67CD4952)
7. MessageType: Reply
8. - ServiceReply:
9. ReplyStatus: Message accepted
10. - MessageAccepted:
11. - Verification:
12. Flavor: No Identity Authentication
13. AuthDataLength: 0 (0x0)
14. AcceptState: Call succeeded
15. - Unm: GETNTCREDSFROMUNIXGROUPNAMEW\_PROC Reply
16. - WindowsCredsW:
17. Status: 0
18. Reserved: Send as 0x00000000
19. - WindowsAccountName: 0x1
20. - UNMWindowsNameW: NFS-DOM-1\g1
21. Length: 24
22. Data: NFS-DOM-1\g1

## GETUNIXCREDSFROMNTGROUPNAMEW\_PROC

The client sends a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) packet to the User Name Mapping Protocol service requesting the [**POSIX**](#gt_b1c15a13-c6a7-40bb-a0db-1947a755118d) group mapping for the Windows group "nfs-dom-1\Domain Admins".

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 2126, Total IP Length = 120
4. + Udp: SrcPort = 965, DstPort = UNM(819), Length = 100
5. - Rpc: Call, Program = mapsvc, Procedure =
6. GETUNIXCREDSFROMNTGROUPNAMEW\_PROC
7. TransactionID: 1758284114 (0x68CD4952)
8. MessageType: Call
9. - ServiceCall:
10. RPCVersionNumber: 2 (0x2)
11. ProgramNumber: mapsvc, 351455(0x00055CDF)
12. ProgramVersion: 2 (0x2)
13. ProcedureNumber: GETUNIXCREDSFROMNTGROUPNAMEW\_PROC
14. - Credential: No Identity Authentication
15. Flavor: No Identity Authentication
16. AuthDataLength: 0 (0x0)
17. - Verification:
18. Flavor: No Identity Authentication
19. AuthDataLength: 0 (0x0)
20. - Unm: GETUNIXCREDSFROMNTGROUPNAMEW\_PROC Call
21. - WindowsGroupAccountNameW:
22. - WindowsAccountName: 0x1
23. - UNMNameW: nfs-dom-1\Domain Admins
24. Length: 46
25. Data: nfs-dom-1\Domain Admins
26. Padding: Binary Large Object (2 Bytes)

The User Name Mapping Protocol service on the server responds with a SUNRPC response packet with the [**group map**](#gt_5501e0d8-1bdc-464c-8c40-c78de7b96734) for Windows group "nfs-dom-1\Domain Adminis" to the POSIX group "bin", illustrated as follows.

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 14900, Total IP Length = 72
4. + Udp: SrcPort = UNM(819), DstPort = 965, Length = 52
5. - Rpc: Reply, Status = Message accepted, Detail = Call succeeded
6. TransactionID: 1758284114 (0x68CD4952)
7. MessageType: Reply
8. - ServiceReply:
9. ReplyStatus: Message accepted
10. - MessageAccepted:
11. - Verification:
12. Flavor: No Identity Authentication
13. AuthDataLength: 0 (0x0)
14. AcceptState: Call succeeded
15. - Unm: GETUNIXCREDSFROMNTGROUPNAMEW\_PROC Reply
16. - UnixCredsW:
17. - UnixAccountName: 0x1
18. - UNMNameW: bin
19. Length: 6
20. Data: bin
21. Padding: Binary Large Object (2 Bytes)
22. ID: 1
23. GidCount: 0

## GETUNIXCREDSFROMNTUSERSIDW\_PROC

The client sends a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) packet to the User Name Mapping Protocol service requesting the [**POSIX**](#gt_b1c15a13-c6a7-40bb-a0db-1947a755118d) credentials for the Windows user [**SID**](#gt_83f2020d-0804-4840-a5ac-e06439d50f8d) "S-1-5-21-3994172400-2625080034-4079281819-500" representing Windows user account "nfs-dom-1\administrator".

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 50594, Total IP Length = 100
4. + Udp: SrcPort = 1013, DstPort = UNM(819), Length = 80
5. - Rpc: Call, Program = mapsvc, Procedure =
6. GETUNIXCREDSFROMNTUSERSIDW\_PROC
7. TransactionID: 1221456821 (0x48CDF3B5)
8. MessageType: Call
9. - ServiceCall:
10. RPCVersionNumber: 2 (0x2)
11. ProgramNumber: mapsvc, 351455(0x00055CDF)
12. ProgramVersion: 2 (0x2)
13. ProcedureNumber: GETUNIXCREDSFROMNTUSERSIDW\_PROC
14. - Credential: No Identity Authentication
15. Flavor: No Identity Authentication
16. AuthDataLength: 0 (0x0)
17. - Verification:
18. Flavor: No Identity Authentication
19. AuthDataLength: 0 (0x0)
20. - Unm: GETUNIXCREDSFROMNTGROUPNAMEW\_PROC Call
21. - WindowsUserAccountNameW:
22. - WindowsAccountName: 0x1
23. - UNMNameW:
24. Length: 28
25. Data: 01 05 00 00 00 00 00 05 15 00 00 00 F0 3B 12 EE
26. E2 8A 77 9C 9B E6 24 F3 F4 01 00 00

The User Name Mapping Protocol service on the server responds with a SUNRPC response packet with the POSIX credentials for the mapped [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account corresponding to Windows user "nfs-dom-1\Administrator" as POSIX user "root", illustrated as follows.

1. Frame:
2. + Ethernet: Etype = Internet IP (IPv4)
3. + Ipv4: Next Protocol = UDP, Packet ID = 13116, Total IP Length = 80
4. + Udp: SrcPort = UNM(819), DstPort = 1013, Length = 60
5. - Rpc: Reply, Status = Message accepted, Detail = Call succeeded
6. TransactionID: 1221456821 (0x48CDF3B5)
7. MessageType: Reply
8. - ServiceReply:
9. ReplyStatus: Message accepted
10. - MessageAccepted:
11. - Verification:
12. Flavor: No Identity Authentication
13. AuthDataLength: 0 (0x0)
14. AcceptState: Call succeeded
15. - Unm: GETUNIXCREDSFROMNTGROUPNAMEW\_PROC Reply
16. - UnixCredsW:
17. - UnixAccountName: 0x1
18. - UNMNameW: root
19. Length: 8
20. Data: root
21. ID: 0
22. GidCount: 2
23. - GID:
24. GID: 1
25. GID: 1

# Security

The User Name Mapping Protocol accepts requests with [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) authentication level AUTH\_NULL.

## Security Considerations for Implementers

None.

## Index of Security Parameters

None.

# Appendix A: Full SunRPC IDL

This IDL section excludes the following procedures, which need to be coded separately because the IDL is unable to describe the returned data types. Sample code for the required structure definitions and encode/decode routines can be found in section [7](#Section_27b23be1334c49448175ec784cfcfb29).

Version 1

* DUMPALLMAPS\_PROC (procedure 4)
* DUMPALLMAPSEX\_PROC (procedure 6)

Version 2

* DUMPALLMAPS\_PROC (procedure 4)
* DUMPALLMAPSEX\_PROC (procedure 6)
* DUMPALLMAPSW\_PROC (procedure 10)
* DUMPALLMAPSEXW\_PROC (procedure 11)
1. const MAXNAMELEN = 128;
2. const MAXNAMELENx2 = 256;
3. const MAXLINELEN = 256;
4. const MAXLINELENx2 = 512;
5. const MAXGIDS = 32;
6. const MAXSIDLEN = 72;
7. typedef opaque MapSvrMBCSNameString<MAXNAMELEN>;
8. typedef opaque MapSvrUnicodeNameString<MAXNAMELENx2>;
9. typedef opaque MapSvrMBCSWindowsNameString<MAXLINELEN>;
10. typedef opaque MapSvrUnicodeWindowsNameString<MAXLINELENx2>;
11. typedef opaque MapSvrMBCSMapString<MAXLINELEN>;
12. typedef opaque MapSvrUnicodeMapString<MAXLINELENx2>;
13. struct unix\_account {
14. long SearchOption;
15. long Reserved;
16. long ID;
17. MapSvrMBCSNameString UnixAccountName;
18. };
19. struct unix\_accountW {
20. long SearchOption;
21. long Reserved;
22. long ID;
23. MapSvrUnicodeNameString UnixAccountName;
24. };
25. struct unix\_user\_auth {
26. MapSvrMBCSNameString UnixUserAccountName;
27. MapSvrMBCSNameString UnixUserAccountPassword;
28. };
29. struct unix\_user\_authW {
30. MapSvrUnicodeNameString UnixUserAccountName;
31. MapSvrUnicodeNameString UnixUserAccountPassword;
32. };
33. struct windows\_creds {
34. long Status;
35. long Reserved;
36. MapSvrMBCSWindowsNameString WindowsAccountName;
37. };
38. struct windows\_credsW {
39. long Status;
40. long Reserved;
41. MapSvrUnicodeWindowsNameString WindowsAccountName;
42. };
43. struct windows\_account {
44. MapSvrMBCSNameString WindowsAccountName;
45. };
46. struct windows\_accountW {
47. MapSvrUnicodeNameString WindowsAccountName;
48. };
49. struct unix\_auth {
50. MapSvrMBCSNameString UnixAccountPassword;
51. long ID;
52. long GIDArray<MAXGIDS>;
53. };
54. struct unix\_authW {
55. MapSvrUnicodeNameString UnixAccountPassword;
56. long ID;
57. long GIDArray<MAXGIDS>;
58. };
59. struct unix\_creds {
60. MapSvrMBCSNameString UnixAccountName;
61. long ID;
62. long GIDArray<MAXGIDS>;
63. };
64. struct unix\_credsW {
65. MapSvrUnicodeNameString UnixAccountName;
66. long ID;
67. long GIDArray<MAXGIDS>;
68. };
69. struct dump\_map\_req {
70. long PrincipalType;
71. long MapRecordIndex;
72. };
73. struct sequence\_number {
74. long CurrentVersionTokenLowPart;
75. long CurrentVersionTokenHighPart;
76. };
77. struct mapping\_record {
78. MapSvrMBCSNameString WindowsAccountName;
79. MapSvrMBCSNameString UnixAccountName;
80. long ID;
81. };
82. struct sid {
83. char SID<MAXSIDLEN>;
84. };
85. struct mapping\_recordW {
86. MapSvrUnicodeNameString WindowsAccountName;
87. MapSvrUnicodeNameString UnixAccountName;
88. long ID;
89. };
90. program MAPPROG {
91. version MAPVERS\_V1 {
92. void
93. MAPPROC\_NULL(void) = 0;
94. windows\_creds
95. GETWINDOWSCREDSFROMUNIXUSERNAME\_PROC(unix\_account)= 1;
96. unix\_creds
97. GETUNIXCREDSFROMNTUSERNAME\_PROC(windows\_account) = 2;
98. unix\_auth
99. AUTHUSINGUNIXCREDS\_PROC(unix\_user\_auth) = 3;
100. sequence\_number
101. GETCURRENTVERSIONTOKEN\_PROC(sequence\_number) = 5;
102. windows\_creds
103. GETWINDOWSGROUPFROMUNIXGROUPNAME\_PROC(unix\_account)= 7;
104. unix\_creds
105. GETUNIXCREDSFROMNTGROUPNAME\_PROC(windows\_account) = 8;
106. } = 1;
107. } = 351455;
108. program MAPPROG {
109. version MAPVERS\_V2 {
110. void
111. MAPPROC\_NULL(void) = 0;
112. windows\_creds
113. GETWINDOWSCREDSFROMUNIXUSERNAME\_PROC(unix\_account)= 1;
114. unix\_creds
115. GETUNIXCREDSFROMNTUSERNAME\_PROC(windows\_account) = 2;
116. unix\_auth
117. AUTHUSINGUNIXCREDS\_PROC(unix\_user\_auth) = 3;
118. sequence\_number
119. GETCURRENTVERSIONTOKEN\_PROC(sequence\_number) = 5;
120. windows\_creds
121. GETWINDOWSGROUPFROMUNIXGROUPNAME\_PROC(unix\_account)= 7;
122. unix\_creds
123. GETUNIXCREDSFROMNTGROUPNAME\_PROC(windows\_account) = 8;
124. unix\_creds
125. GETUNIXCREDSFROMNTUSERSID\_PROC(sid) = 9;
126. windows\_credsW
127. GETWINDOWSUSERFROMUNIXUSERNAMEW\_PROC(unix\_accountW)=12;
128. unix\_credsW
129. GETUNIXCREDSFROMNTUSERNAMEW\_PROC(windows\_accountW)= 13;
130. unix\_authW
131. AUTHUSINGUNIXCREDSW\_PROC(unix\_user\_authW) = 14;
132. windows\_credsW
133. GETWINDOWSGROUPFROMUNIXGROUPNAMEW\_PROC(unix\_accountW)= 15;
134. unix\_credsW
135. GETUNIXCREDSFROMNTGROUPNAMEW\_PROC(windows\_accountW) = 16;
136. unix\_credsW
137. GETUNIXCREDSFROMNTUSERSIDW\_PROC(sid) = 17;
138. } = 2;
139. } = 351455;

# Appendix B: Sample Code to Encode and Decode Non-XDR-Compliant Data Types

Interpret the sample code in the following sections as being written in the C programming language.

## Header File Content

1. struct mapping {
2. sequence\_number Token;
3. uint MappingRecordCount;
4. uint TotalMappingRecordCount;
5. mapping\_record \*MapArray;
6. };
7. typedef struct mapping mapping;
8. struct maps {
9. sequence\_number Token;
10. uint MappingRecordCount;
11. uint TotalMappingRecordCount;
12. MapSvrMBCSMapString \*MapArray;
13. };
14. typedef struct maps maps;
15.
16. struct mappingW {
17. sequence\_number Token;
18. uint MappingRecordCount;
19. uint TotalMappingRecordCount;
20. mapping\_recordW \*MapArray;
21. };
22. typedef struct mappingW mappingW;
23.
24. struct mapsW {
25. sequence\_number Token;
26. uint MappingRecordCount;
27. uint TotalMappingRecordCount;
28. MapSvrUnicodeMapString \*MapArray;
29. };
30. typedef struct mapsW mapsW;
31. #define DUMPALLMAPS\_PROC 4
32. extern mapping \* dumpallmaps\_proc\_1(dump\_map\_req \*, CLIENT \*);
33. extern mapping \* dumpallmaps\_proc\_1\_svc(dump\_map\_req \*,
34. struct svc\_req \*);
35. #define DUMPALLMAPSEX\_PROC 6
36. extern maps \* dumpallmapsex\_proc\_1(dump\_map\_req \*, CLIENT \*);
37. extern maps \* dumpallmapsex\_proc\_1\_svc(dump\_map\_req \*,
38. struct svc\_req \*);
39. #define DUMPALLMAPS\_PROC 4
40. extern mapping \* dumpallmaps\_proc\_2(dump\_map\_req \*, CLIENT \*);
41. extern mapping \* dumpallmaps\_proc\_2\_svc(dump\_map\_req \*,
42. struct svc\_req \*);
43. #define DUMPALLMAPSEX\_PROC 6
44. extern maps \* dumpallmapsex\_proc\_2(dump\_map\_req \*, CLIENT \*);
45. extern maps \* dumpallmapsex\_proc\_2\_svc(dump\_map\_req \*,
46. struct svc\_req \*);
47.
48. #define DUMPALLMAPSW\_PROC 10
49. extern mappingW \* dumpallmapsw\_proc\_2(dump\_map\_req \*, CLIENT \*);
50. extern mappingW \* dumpallmapsw\_proc\_2\_svc(dump\_map\_req \*,
51. struct svc\_req \*);
52.
53. #define DUMPALLMAPSEXW\_PROC 11
54. extern mapsW \* dumpallmapsexw\_proc\_2(dump\_map\_req \*, CLIENT \*);
55. extern mapsW \* dumpallmapsexw\_proc\_2\_svc(dump\_map\_req \*,
56. struct svc\_req \*);
57. extern bool\_t xdr\_mapping(XDR \*, mapping\*);
58. extern bool\_t xdr\_maps(XDR \*, maps\*);
59. extern bool\_t xdr\_mappingW(XDR \*, mappingW\*);
60. extern bool\_t xdr\_mapsW(XDR \*, mapsW\*);

## Encode/Decode Routines For Non-XDR Data Types Using XDR Primitives

1. bool\_t
2. xdr\_mapping(register XDR \*xdrs, mapping \*objp)
3. {
4. if (!xdr\_sequence\_number(xdrs, &objp->Token))
5. return (FALSE);
6. if (!xdr\_u\_int(xdrs, &objp->MappingRecordCount))
7. return (FALSE);
8. if (!xdr\_u\_int(xdrs, &objp->TotalMappingRecordCount))
9. return (FALSE);
10. objp->MapArray = (mapping\_record \*) malloc (
11. objp->MappingRecordCount \* sizeof (mapping\_record));
12.
13. if (!objp->MapArray)
14. return (FALSE);
15.
16. if (!xdr\_vector(xdrs,
17. (char \*)objp->MapArray,
18. objp->MappingRecordCount,
19. sizeof (mapping\_record),
20. (xdrproc\_t) xdr\_mapping\_record))
21. return (TRUE);
22. }
23. bool\_t
24. xdr\_maps(register XDR \*xdrs, maps \*objp)
25. {
26. if (!xdr\_sequence\_number(xdrs, &objp->Token))
27. return (FALSE);
28. if (!xdr\_u\_int(xdrs, &objp->MappingRecordCount))
29. return (FALSE);
30. if (!xdr\_u\_int(xdrs, &objp->TotalMappingRecordCount))
31. return (FALSE);
32.
33. objp->MapArray = (MapSvrMBCSMapString \*) malloc (
34. objp->MappingRecordCount \* sizeof (MapSvrMBCSMapString));
35.
36. if (!objp->MapArray)
37. return (FALSE);
38.
39. if (!xdr\_vector(xdrs,
40. (char \*)objp->MapArray,
41. objp->MappingRecordCount,
42. sizeof (MapSvrMBCSMapString),
43. (xdrproc\_t) xdr\_MapSvrMBCSMapString))
44. return (FALSE);
45. return (TRUE);
46. }
47. bool\_t
48. xdr\_mappingW(register XDR \*xdrs, mappingW \*objp)
49. {
50. if (!xdr\_sequence\_number(xdrs, &objp->Token))
51. return (FALSE);
52. if (!xdr\_u\_int(xdrs, &objp->MappingRecordCount))
53. return (FALSE);
54. if (!xdr\_u\_int(xdrs, &objp->TotalMappingRecordCount))
55. return (FALSE);
56.
57. objp->MapArray = (mapping\_recordW \*) malloc (
58. objp->MappingRecordCount \* sizeof (mapping\_recordW));
59. if (!objp->MapArray)
60. return (FALSE);
61. if (!xdr\_vector(xdrs,
62. (char \*)objp->MapArray,
63. objp->MappingRecordCount,
64. sizeof (mapping\_recordW),
65. (xdrproc\_t) xdr\_mapping\_recordW))
66. return (FALSE);
67. return (TRUE);
68. }
69. bool\_t
70. xdr\_mapsW(register XDR \*xdrs, mapsW \*objp)
71. {
72. if (!xdr\_sequence\_number(xdrs, &objp->Token))
73. return (FALSE);
74. if (!xdr\_u\_int(xdrs, &objp->MappingRecordCount))
75. return (FALSE);
76. if (!xdr\_u\_int(xdrs, &objp->TotalMappingRecordCount))
77. return (FALSE);
78. objp->MapArray = (MapSvrUnicodeMapString \*) malloc (
79. objp->MappingRecordCount \* sizeof (MapSvrUnicodeMapString));
80. if (!objp->MapArray)
81. return (FALSE);
82. if (!xdr\_vector(xdrs,
83. (char \*)objp->MapArray,
84. objp->MappingRecordCount,
85. sizeof (MapSvrUnicodeMapString),
86. (xdrproc\_t) xdr\_MapSvrUnicodeMapString))
87. return (FALSE);
88.
89. return (TRUE);
90. }

# Appendix C: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include updates to those products.

* Windows Server 2003 R2 operating system
* Windows Vista operating system
* Windows Server 2008 operating system
* Windows 7 operating system
* Windows Server 2008 R2 operating system
* Windows 8 operating system
* Windows Server 2012 operating system
* Windows 8.1 operating system
* Windows Server 2012 R2 operating system
* Windows 10 operating system
* Windows Server 2016 operating system
* Windows Server operating system

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms "SHOULD" or "SHOULD NOT" implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term "MAY" implies that the product does not follow the prescription.

[<1> Section 1.4](#Appendix_A_Target_1): The Windows implementation of the User Name Mapping Protocol server is used by the "Client for NFS", "Server for NFS", and "Remote Shell Service" components of the "Services for UNIX" product suite and Windows Server 2003 R2.

[<2> Section 1.5](#Appendix_A_Target_2): [[NFSAUTH]](https://go.microsoft.com/fwlink/?LinkId=90231) describes how the Windows implementation of the User Name Mapping Protocol is configured with appropriate Windows and [**UNIX**](#gt_6084f060-d2f4-4ebb-8679-2d2ffa43b3ed) account mappings.

[<3> Section 1.6](#Appendix_A_Target_3): Windows Server 2003 R2 is the only version of Windows with an implementation of the User Name Mapping Protocol server.

[<4> Section 2.2.2.6](#Appendix_A_Target_4): Due to an error in the Windows Server 2003 R2 implementation of the User Name Mapping Protocol server, for maps obtained from an NIS service, the value of the **AuthType** field can unpredictably be set to '0' (AUTH\_FILE) rather than the expected value of '1' (AUTH\_NIS). The other fields in the map are valid.

**Note**  Windows Server 2003 R2 is the only version of Windows with an implementation of the User Name Mapping Protocol server.

[<5> Section 2.2.4](#Appendix_A_Target_5): Although the only authentication mechanism defined by the User Name Mapping Protocol is AUTH\_NULL (as specified in [[RFC1057]](https://go.microsoft.com/fwlink/?LinkId=90265) section 9.1), the Windows implementation of the User Name Mapping Protocol server returns a [**SUNRPC**](#gt_bf3dc49e-0dc7-47d3-a0c9-a0a57656c600) status of MSG\_DENIED with a reject status of AUTH\_ERROR, and an authentication status of AUTH\_BADCRED, whenever the client IP address does not match the list of trusted client addresses as configured by the administrator.

[<6> Section 2.2.5.4](#Appendix_A_Target_6): The Windows implementation of the User Name Mapping Protocol server uses only the UNIX user name as search criteria; the password given as input is ignored.

[<7> Section 2.2.5.15](#Appendix_A_Target_7): The Windows implementation of the User Name Mapping Protocol server uses only the UNIX user name as search criteria; the password given as input is ignored.

[<8> Section 3.2.5.1](#Appendix_A_Target_8): The Windows implementation of the User Name Mapping Protocol server returns a SUNRPC status of MSG\_DENIED with a reject status of AUTH\_ERROR and a authentication status of AUTH\_BADCRED, whenever the client IP address does not match the list of trusted client addresses, as configured by the administrator. This Windows-specific behavior is documented in [NFSAUTH].

[<9> Section 3.2.5.2.1](#Appendix_A_Target_9): The Windows implementation of the User Name Mapping Protocol server returns a maximum of 200 mapping records in a SUNRPC packet response.

[<10> Section 3.2.5.2.1](#Appendix_A_Target_10): The Windows implementation of the User Name Mapping Protocol initializes **CurrentVersionTokenHighPart** and **CurrentVersionTokenLowPart** to a locally unique identifier as returned by the Win32 API AllocateLocallyUniqueId() on startup.

[<11> Section 3.2.5.2.2](#Appendix_A_Target_11): The Windows implementation of the User Name Mapping Protocol server limits the number of mapping records returned by both DUMPALLMAPS\_PROC and DUMPALLMAPSW\_PROC in a single SUNRPC packet response to a value set by the registry DWORD value **HKLM\System\CurrentControlSet\Services\MapSvc\CurrentVersion\WriteBlock**. The default value for this limit is 200. Any changes to this value require a restart of the Windows implementation of the User Name Mapping Protocol server in order to become effective.

[<12> Section 3.2.5.2.2](#Appendix_A_Target_12): The Windows implementation of the User Name Mapping Protocol server limits the size of a single SUNRPC response that uses the UDP transport to 8,800 bytes. There is no such limit when using the TCP transport. For the [**DUMPALLMAPSXXXX\_PROC**](#gt_a85344db-a5da-41e9-a60c-36d344a282b5) procedures, if the number of records to be returned in a single SUNRPC UDP response message cannot be contained in a message of this size, then the Windows implementation of the User Name Mapping Protocol server replies with a SUNRPC message status of MSG\_ACCEPTED, with the ACCEPT status set to SYSTEM\_ERR.

[<13> Section 3.2.5.2.2](#Appendix_A_Target_13): The Windows implementation of the User Name Mapping Protocol server ignores the value of *MapRecordIndex* in the DUMPALLMAPSEX\_PROC and DUMPALLMAPSEXW\_PROC request, and the response always contains the maps enumerated from index 0. In the response, the **TotalMappingRecordCount** value is set to the sum of the total number of map records held by the server and the value of *MapRecordIndex* in the request.

[<14> Section 3.2.5.2.3](#Appendix_A_Target_14): The Windows implementation of the User Name Mapping Protocol server changes the 64-bit integer sequence number to a random value every time the account mapping database is updated. This value is unique only within the lifetime of the current server process—its uniqueness can only be guaranteed within the span of a single server process. The server returns this sequence number to the client as **CurrentVersionTokenHighPart** and **CurrentVersionTokenLowPart** in the [GETCURRENTVERSIONTOKEN\_PROC](#Section_5ec438be26d345809cd3ad4bab44c9e9) response and the DUMPALLMAPSXXX\_PROC response.

# Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as Major, Minor, or None.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

* A document revision that incorporates changes to interoperability requirements.
* A document revision that captures changes to protocol functionality.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **None** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

| Section | Description | Revision class |
| --- | --- | --- |
| [8](#Section_2bfcb4f945174a149d32eed1d26edf84) Appendix C: Product Behavior | Added Windows Server to the applicable products list. | Major |

# Index

A

Abstract data model

 [client](#section_cf8fac4a0f6b4759abe0a5e38bae1180) 33

 [server](#section_951ed3deacce4d4aa2be27c1b4428c06) 36

[Applicability](#section_3e0cb189b7804d7d8ed8dffa99103c60) 11

[AUTHUSINGUNIXCREDS\_PROC](#section_fb03e47942d941088b7ae39e3235ae2f) 41

[AUTHUSINGUNIXCREDS\_PROC (PROC 3)](#section_c7f40a27eb2249d7a1d77e428bc61eb9) 26

[AUTHUSINGUNIXCREDSW\_PROC](#section_1484aaae4317475f936b0b541527d7f0) 55

[AUTHUSINGUNIXCREDSW\_PROC (PROC 14)](#section_f80f8d46fca64f58b7e06e7f478eb121) 31

C

[Capability negotiation](#section_352383b4d2da428eb497e60e3b86d0af) 11

[Change tracking](#section_d907743730bd4f298173ba9c26d198f1) 69

Client

 [abstract data model](#section_cf8fac4a0f6b4759abe0a5e38bae1180) 33

 [higher-layer triggered events](#section_9d4e840f2ec04435a05fcd80e309cb99) 34

 [initialization](#section_4d68133e3c7740cd85b8da54466f8e6f) 34

 [local events](#section_51e0d95ecfd743069c61aad064205a82) 36

 [message processing](#section_dda41891285340d9a0dd853d8e275e87) 34

 [sequencing rules](#section_dda41891285340d9a0dd853d8e275e87) 34

 [timer events](#section_1d4eb8d202924dfaa882b677e1b3a38a) 36

 [timers](#section_04be40bb941b4baa80276dbecc9dd92f) 34

[Common User Name Mapping Protocol Data Types](#section_04730b74ae684f15a98fff38f6dc32b7) 12

[Common User Name Mapping Protocol Data Types message](#section_04730b74ae684f15a98fff38f6dc32b7) 12

D

Data model - abstract

 [client](#section_cf8fac4a0f6b4759abe0a5e38bae1180) 33

 [server](#section_951ed3deacce4d4aa2be27c1b4428c06) 36

[dump\_map\_req](#section_169ff18bd16740b7b79298744e8c7bff) 20

[DUMPALLMAPS\_PROC](#section_2c10d52621c947ffae270f883783c7cc) 42

[DUMPALLMAPS\_PROC (PROC 4)](#section_50e135e64d374c85ac7d845ba95f6516) 27

[DUMPALLMAPSEX\_PROC](#section_57965883bc534f948e8fafc881b76bc0) 45

[DUMPALLMAPSEX\_PROC (PROC 6)](#section_a905c191dcce4c1187569c717a726cbf) 28

[DUMPALLMAPSEXW\_PROC](#section_e36752cc2a4747d7af77e5cacb0392c1) 52

[DUMPALLMAPSEXW\_PROC (PROC 11)](#section_66b7197697114ce899f841b062e9e07f) 29

[DUMPALLMAPSW\_PROC](#section_b6a6ae78655b4078813aa622e56e7f92) 50

[DUMPALLMAPSW\_PROC (PROC 10)](#section_a06afaccc7194e83956c605013d857b0) 29

[DUMPALLMAPSXXX\_PROC request](#section_a24000c7b2b340ad90f717ad1ab8c95c) 37

F

[Fields - vendor-extensible](#section_18820d94cd2849ecb6c44bfa64db34cc) 11

[Full SunRPC IDL](#section_e2c4679291534d58ba9bb5b324df943e) 61

G

[GETCURRENTVERSIONTOKEN\_PROC](#section_5ac3d3aab3864441a45df8f79e7576e1) 45

[GETCURRENTVERSIONTOKEN\_PROC (PROC 5)](#section_5ec438be26d345809cd3ad4bab44c9e9) 27

[GETCURRENTVERSIONTOKEN\_PROC request](#section_a24000c7b2b340ad90f717ad1ab8c95c) 37

[GETUNIXCREDSFROMNTGROUPNAME\_PROC](#section_9261fea310b4419a8cec5444c38ce721) 48

[GETUNIXCREDSFROMNTGROUPNAME\_PROC (PROC 8)](#section_0d2556076a2647a3855bb1790221e020) 28

[GETUNIXCREDSFROMNTGROUPNAMEW\_PROC](#section_7039e78a7ffa4607bfdfb82d3d306c24) 57

[GETUNIXCREDSFROMNTGROUPNAMEW\_PROC (PROC 16)](#section_d3649cf4fe6a4681b8d5879f75de6436) 31

[GETUNIXCREDSFROMNTUSERNAME\_PROC](#section_d3db6a456b08413da6fda2f66d6b1fca) 40

[GETUNIXCREDSFROMNTUSERNAME\_PROC (PROC 2)s](#section_066f33191f294f649f07edb337e90d32) 26

[GETUNIXCREDSFROMNTUSERNAMEW\_PROC](#section_7a357d1059ca478883a9aabc3b66ce77) 54

[GETUNIXCREDSFROMNTUSERNAMEW\_PROC (PROC 13)](#section_680ba18727d7457c9bb23b8ca9e7f259) 30

[GETUNIXCREDSFROMNTUSERSID\_PROC](#section_03b1d0ed143e42cd867c08ffaba71307) 49

[GETUNIXCREDSFROMNTUSERSID\_PROC (PROC 9)](#section_bb264bb5231742f181d24c705d29bf15) 29

[GETUNIXCREDSFROMNTUSERSIDW\_PROC](#section_bd6e3f9745234d31a5383fe91571b07f) 58

[GETUNIXCREDSFROMNTUSERSIDW\_PROC (PROC 17)](#section_4f65e9e1c46b43fab3c02537e5766b3b) 32

[GETWINDOWSCREDSFROMUNIXUSERNAME\_PROC](#section_d0bb90039e104eb480063dbe77cd92c3) 39

[GETWINDOWSCREDSFROMUNIXUSERNAME\_PROC (PROC 1)](#section_887248fa50f84268a47d27583ff97cf8) 26

[GETWINDOWSGROUPFROMUNIXGROUPNAME\_PROC](#section_0abbf40b1deb4a249c06127d8484c2cb) 47

[GETWINDOWSGROUPFROMUNIXGROUPNAME\_PROC (PROC 7)](#section_ef1fd6724c4646d0b3b779c2fa8e62e9) 28

[GETWINDOWSGROUPFROMUNIXGROUPNAMEW\_PROC](#section_8dc553f7cc474eba958427173ac519a9) 56

[GETWINDOWSGROUPFROMUNIXGROUPNAMEW\_PROC (PROC 15)](#section_7e7e7c29834f4f6ba651fa1388e7de0a) 31

[GETWINDOWSUSERFROMUNIXUSERNAMEW\_PROC](#section_f279d38319c64b02892058f58e1ef785) 53

[GETWINDOWSUSERFROMUNIXUSERNAMEW\_PROC (PROC 12)](#section_45e29142f3e247a19cad5d4758318bd4) 30

[Glossary](#section_058e9e924df74a35a3f056df6249c98d) 7

H

Higher-layer triggered events

 [client](#section_9d4e840f2ec04435a05fcd80e309cb99) 34

 [server](#section_53443caf4057491dbbe0de693dffcfa1) 37

I

[IDL](#section_e2c4679291534d58ba9bb5b324df943e) 61

[Implementer - security considerations](#section_da7da1c6f8e547eb98e26550527f53c4) 60

[Index of security parameters](#section_4bc227b8d917448eb3d40be8f1c6eafe) 60

[Informative references](#section_bd9860b8658141dd887f0c7fc834529b) 9

Initialization

 [client](#section_4d68133e3c7740cd85b8da54466f8e6f) 34

 [server](#section_e7eed48de6414299ae4c05019107f6ef) 37

[Introduction](#section_800fc5dbd5ef4d93b88f576bff153916) 7

L

Local events

 [client](#section_51e0d95ecfd743069c61aad064205a82) 36

 [server](#section_dec6e11bc7144d2c856eb8a7b9febb88) 38

M

[mapping](#section_985b1e0747624be3849a9fb2dc6f1a8c) 22

[mapping\_record](#section_8153cbb8f0ea4c6d9a99688ed10a3c5a) 21

[mapping\_recordW](#section_d301c215b147412cb44b60e519d1b4e7) 22

[mappingW](#section_dd0096a2f16f46dca8ab0d18ad292a44) 23

[MAPPROC\_NULL (PROC 0)](#section_4530278cc7684666af39e2684f7ef48a) 25

[maps](#section_07e03281db04473588930136b435ddb4) 23

[MapSvrMBCSMapString](#section_d5e345407d2e42a3856a7ac74d3750e5) 13

[MapSvrMBCSNameString](#section_ff8f275537504d02be9ce4e3b1619b5f) 13

[MapSvrMBCSWindowsNameString](#section_86c54411d47d4ff8993d8bf7d3111717) 13

[MapSvrUnicodeMapString](#section_ce09d111bdc247eab853ff417401c7d6) 15

[MapSvrUnicodeNameString](#section_dbd5ac38df0c4798a86ead90687b0ca2) 13

[MapSvrUnicodeWindowsNameString](#section_2cab5dbd18cb4ce8b416ae5b23b994ba) 13

[mapsW](#section_003667ced8bb4e5eafb6a73e3340f67a) 23

Message processing

 [client](#section_dda41891285340d9a0dd853d8e275e87) 34

 [server](#section_d24c6377586841dd8b9786b7c0c84091) 37

Messages

 [Common User Name Mapping Protocol Data Types](#section_04730b74ae684f15a98fff38f6dc32b7) 12

 [Non-XDR-Compliant Data Structures](#section_5a9a16bf9a66408588f7aa4647da4472) 22

 [Standard Failure Responses](#section_5504ab97ff774106aa677e8353d09fbb) 24

 [syntax](#section_3677a9ee8f1c4ef6809f725a3154ce0b) 12

 [transport](#section_398135ed017c4bd991aebeb429088c99) 12

 [User Name Mapping Protocol Messages](#section_f29045a2a1c54fa9a3c96c2e35b2f1ae) 25

N

[Non-XDR-Compliant Data Structures message](#section_5a9a16bf9a66408588f7aa4647da4472) 22

[Normative references](#section_f933f1ad69b540f384ee49c147081bb9) 9

O

Other local events

 [server](#section_dec6e11bc7144d2c856eb8a7b9febb88) 38

[Overview (synopsis)](#section_70e6de24bbae4ee7985305d69198a94c) 9

P

[Parameters - security index](#section_4bc227b8d917448eb3d40be8f1c6eafe) 60

[Polling for cache consistency](#section_d1074c84f93748d290900e78db0cd921) 35

[Preconditions](#section_5a5d8aa446c54f58a2494283324bec62) 11

[Prerequisites](#section_5a5d8aa446c54f58a2494283324bec62) 11

[Processing for all procedures](#section_d6f52270799d4186a7129f0cd91ff030) 37

[Product behavior](#section_2bfcb4f945174a149d32eed1d26edf84) 67

Protocol Details

 [overview](#section_975f1b23f84d475684588e4ea9606cdf) 33

R

[References](#section_9a14ac5dd9b3444497e17f094a9bee3f) 9

 [informative](#section_bd9860b8658141dd887f0c7fc834529b) 9

 [normative](#section_f933f1ad69b540f384ee49c147081bb9) 9

[Relationship to other protocols](#section_c3911ff16eee4a609038191726498bdc) 10

[Request from the server - processing account mapping response](#section_4c212db1c7b94622a51697d0951c58b0) 35

Request to the server

 [further account mapping](#section_c5b938b777984547ba267c687c033b2c) 35

 [initial account mapping](#section_6b0ed9ef75f849c6a96daa984aa47d47) 35

S

Security

 [implementer considerations](#section_da7da1c6f8e547eb98e26550527f53c4) 60

 [overview](#section_167073e7393e4e78916e81c25a6fb0f5) 60

 [parameter index](#section_4bc227b8d917448eb3d40be8f1c6eafe) 60

[sequence\_number](#section_6b538aa7a21543c8876f93c409d3c5f3) 20

Sequencing rules

 [client](#section_dda41891285340d9a0dd853d8e275e87) 34

 [server](#section_d24c6377586841dd8b9786b7c0c84091) 37

Server

 [abstract data model](#section_951ed3deacce4d4aa2be27c1b4428c06) 36

 [higher-layer triggered events](#section_53443caf4057491dbbe0de693dffcfa1) 37

 [initialization](#section_e7eed48de6414299ae4c05019107f6ef) 37

 [local events](#section_dec6e11bc7144d2c856eb8a7b9febb88) 38

 [message processing](#section_d24c6377586841dd8b9786b7c0c84091) 37

 [other local events](#section_dec6e11bc7144d2c856eb8a7b9febb88) 38

 [sequencing rules](#section_d24c6377586841dd8b9786b7c0c84091) 37

 [timer events](#section_3f80da6832d14b37a65612650a4c6e74) 38

 [timers](#section_1d3eb7c507974a64a5a6f7ec77331036) 37

[sid](#section_e968eb1d8e3d4c7f97f7d6115a21b0c9) 21

[Sizes](#section_618bd83c71cd432e837459d98fc0a872) 12

[Standard failure responses](#section_5504ab97ff774106aa677e8353d09fbb) 24

[Standard Failure Responses message](#section_5504ab97ff774106aa677e8353d09fbb) 24

[Standards assignments](#section_088abb5390f4443490ef97bc1d65306d) 11

[SunRPC IDL](#section_e2c4679291534d58ba9bb5b324df943e) 61

[SUNRPC Request header](#section_9a65b636ec864dd19ee33d06159e1001) 12

[SUNRPC Response header](#section_c80fe1880271486fb4fdcf342c0f6e96) 12

[Syntax - message](#section_3677a9ee8f1c4ef6809f725a3154ce0b) 12

T

Timer events

 [client](#section_1d4eb8d202924dfaa882b677e1b3a38a) 36

 [server](#section_3f80da6832d14b37a65612650a4c6e74) 38

Timers

 [client](#section_04be40bb941b4baa80276dbecc9dd92f) 34

 [server](#section_1d3eb7c507974a64a5a6f7ec77331036) 37

[Tracking changes](#section_d907743730bd4f298173ba9c26d198f1) 69

[Transport](#section_398135ed017c4bd991aebeb429088c99) 12

[Transport - message](#section_398135ed017c4bd991aebeb429088c99) 12

Triggered events - higher-layer

 [client](#section_9d4e840f2ec04435a05fcd80e309cb99) 34

 [server](#section_53443caf4057491dbbe0de693dffcfa1) 37

U

[unix\_account](#section_b230500c731a4a068b1affe13434f9d7) 15

[unix\_accountW](#section_70e6d2279eaf4e4489476e16bbdb23ec) 16

[unix\_auth](#section_1fa00f3da1fe45949c2c7530c17b20c7) 19

[unix\_authW](#section_0085175ecf324007b4ee997f29e807b3) 19

[unix\_creds](#section_d63527bebf8a44cb81a7ccfbeb7e074e) 19

[unix\_credsW](#section_af4ae7269e04495f92fab22931ebb979) 20

[unix\_user\_auth](#section_1a19252277894a8ca84860d2e2d9e4c9) 16

[unix\_user\_authW](#section_cb91528f066a4e76a3fc541f243e1008) 17

[User Name Mapping Protocol Messages](#section_f29045a2a1c54fa9a3c96c2e35b2f1ae) 25

[User Name Mapping Protocol Messages message](#section_f29045a2a1c54fa9a3c96c2e35b2f1ae) 25

V

[Vendor-extensible fields](#section_18820d94cd2849ecb6c44bfa64db34cc) 11

[Versioning](#section_352383b4d2da428eb497e60e3b86d0af) 11

W

[windows\_account](#section_99f8e7fdb66547f78046bb9dfd28291b) 18

[windows\_accountW](#section_13f138f133804bca923f6da2175e35ed) 18

[windows\_creds](#section_ae9562678a7f4567866196b64aca3367) 17

[windows\_credsW](#section_165f85c0c7f64ddbadc488e2ed0e772f) 18