[MS-RAA-Diff]:

Remote Authorization API 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 or the Microsoft Community Promise. 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.
- **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.
- **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
12/16/2011	1.0	New	Released new document.
3/30/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
7/12/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	1.0	None	No changes to the meaning, language, or formatting of the technical content.
1/31/2013	2.0	Major	Significantly changed the technical content.
8/8/2013	3.0	Major	Significantly changed the technical content.
11/14/2013	4.0	Major	Significantly changed the technical content.
2/13/2014	5.0	Major	Significantly changed the technical content.
5/15/2014	5.0	None	No changes to the meaning, language, or formatting of the technical content.
6/30/2015	6.0	Major	Significantly changed the technical content.
10/16/2015	6.0	None	No changes to the meaning, language, or formatting of the technical content.
7/14/2016	6.0	None	No changes to the meaning, language, or formatting of the technical content.
6/1/2017	6.0	None	No changes to the meaning, language, or formatting of the technical content.
9/15/2017	7.0	Major	Significantly changed the technical content.
9/12/2018	8.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

This document specifies the Remote Authorization API Protocol. The Remote Authorization API Protocol is a Remote Procedure Call (RPC)-based protocol used to perform various authorization queries on remote computers.

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.

1.1 (Updated Section) Glossary

This document uses the following terms:

access control decision: Choosing whether to allow a user access of a resource on a specific remote service based on a given authorization policy.

access control list (ACL): A list of access control entries (ACEs) that collectively describe the security rules for authorizing access to some resource; for example, an object or set of objects.

Active Directory: ~~A~~ The Windows implementation of a general-purpose ~~network~~ directory service~~. Active Directory also refers to the Windows implementation of a directory service.~~, which uses LDAP as its primary access protocol. Active Directory stores information about a variety of objects in the network~~. User~~ such as user accounts, computer accounts, groups, and all related credential information used by ~~the Windows implementation of~~ Kerberos ~~are stored in Active Directory.~~[MS-KILE]. Active Directory is either deployed as Active Directory Domain Services (AD DS) or Active Directory Lightweight Directory Services (AD LDS~~). [MS-ADTS] describes both forms. For more information, see [MS-AUTHSOD] section 1.1.1.5.2, Lightweight Directory Access Protocol (LDAP) versions 2 and 3, Kerberos, and DNS~~), which are both described in [MS-ADOD]: Active Directory Protocols Overview.

authorization policy: A set of rules that govern a user's access to one or more resources or classes of resources.

client context: A context describing an execution environment from which an activation request has originated.

impersonation token: A security context, created on the server, that represents the set of rights and privileges of the user. Used to test whether the user is authorized to access a resource.

principal self: A well-known security identifier (SID) used to represent the identity of a security principal when that security principal is also the object that is being protected with a security descriptor. Applicable only to directory objects that are representing security principals, the principal self identifier allows the security descriptor on the directory object to grant specific user rights to the principal itself. As an example, a user object for fred@domain.com might have a security descriptor that allowed principal-self:update-shoe-size. The intent is to allow fred to update his own shoe size. The use of the fixed value SID for principal self prevents every user object from needing a unique security descriptor, thus conserving space in the directory database.

RPC endpoint: A network-specific address of a server process for remote procedure calls (RPCs). The actual name of the RPC endpoint depends on the RPC protocol sequence being used. For example, for the NCACN_IP_TCP RPC protocol sequence an RPC endpoint might be TCP port 1025. For more information, see [C706].

security descriptor: A data structure containing the security information associated with a securable object. A security descriptor identifies an object's owner by its security identifier (SID). If access control is configured for the object, its security descriptor contains a discretionary access control list (DACL) with SIDs for the security principals who are allowed or

denied access. Applications use this structure to set and query an object's security status. The security descriptor is used to guard access to an object as well as to control which type of auditing takes place when the object is accessed. The security descriptor format is specified in [MS-DTYP] section 2.4.6; a string representation of security descriptors, called SDDL, is specified in [MS-DTYP] section 2.5.1.

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

smart card: A portable device that is shaped like a business card and is embedded with a memory chip and either a microprocessor or some non-programmable logic. Smart cards are often used as authentication tokens and for secure key storage. Smart cards used for secure key storage have the ability to perform cryptographic operations with the stored key without allowing the key itself to be read or otherwise extracted from the card.

token: A set of rights and privileges for a given user.

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

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as defined in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 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.

1.2.1 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.

[C706] The Open Group, "DCE 1.1: Remote Procedure Call", C706, August 1997, https://www2.opengroup.org/ogsys/catalog/c706

[MS-DTYP] Microsoft Corporation, "Windows Data Types".

[MS-ERREF] Microsoft Corporation, "Windows Error Codes".

[MS-KILE] Microsoft Corporation, "Kerberos Protocol Extensions".

[MS-LSAT] Microsoft Corporation, "Local Security Authority (Translation Methods) Remote Protocol".

[MS-RPCE] Microsoft Corporation, "Remote Procedure Call Protocol Extensions".

[MS-SFU] Microsoft Corporation, "Kerberos Protocol Extensions: Service for User and Constrained Delegation Protocol".

[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

1.2.2 Informative References

None.

1.3 Overview

The Remote Authorization API (RAZA) protocol is designed to allow applications to simulate an access control decision that would be made when a given principal attempts to access a resource on a remote service that is protected with a given authorization policy. Because these are simulations, they can vary from the actual groups and/or claims in a user's token.

For example, a user can log on with a password, or the user can log on using a smart card (with authentication assurance provisioned). Each type of logon will result in a different kind of impersonation token. Logging on using the password produces an impersonation token with a mapped group or claim; logging on using the smart card produces an impersonation token without a mapped group or claim.

The following are some of the examples of this protocol's applications:

- Simulate the groups and/or claims that a user would have if the user were to authenticate to a remote service.
- Simulate a user's access to a hypothetical resource on a specific remote service that is protected with a given authorization policy.
- Simulate how potential changes to the user's group or claim assignments can affect access to resources on the remote machine.

The RAZA protocol defines client and server protocol roles.<1> A general description of message flow is as follows:

1. The RAZA client initiates a RAZA conversation by issuing a request to a RAZA server to initialize and maintain a resource manager object.
2. The RAZA server listens to an RPC endpoint. When a client makes the preceding request, the RAZA server creates and maintains state for a resource manager object on behalf of the client.
3. The RAZA client can then request creation of a client context for a user by specifying the user's security identifier (SID). After a client context is successfully created on the server, the RAZA client can examine the contents of the client context (for example, the group SIDs and claims within the client context) and/or modify the client context. Additionally, the RAZA client can perform an "AccessCheck" using the client context and a specified security descriptor.

RAZA supports the following method calls to provide clients a way to simulate access control decisions.

- AuthzrFreeContext
- AuthzrInitializeContextFromSid
- AuthzrInitializeCompoundContext
- AuthzrAccessCheck
- AuthzGetInformationFromContext

- AuthzrModifyClaims
- AuthzrModifySids

1.4 Relationship to Other Protocols

The Remote Authorization API Protocol is dependent on RPC and TCP for its transport.

No other protocol currently depends on the Remote Authorization API Protocol.

1.5 Prerequisites/Preconditions

The Remote Authorization API Protocol is an RPC interface and, as a result, has the prerequisites specified in [MS-RPCE] (section 1.5) as being common to RPC interfaces.

It is assumed that a Remote Authorization API Protocol client has obtained the name of a remote computer that supports the Remote Authorization API Protocol before this protocol is invoked.

1.6 Applicability Statement

This protocol is appropriate only for implementing a tool to remotely approximate and profile "what-if" authorization decisions.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

- **Supported Transports:** This protocol uses multiple RPC Protocol Sequences as specified in section 2.1.
- **Security and Authentication Methods:** As specified in [MS-RPCE] section 3.2.1.4.1.
- **Capability Negotiation:** The RAZA protocol does not support negotiation of the interface version to use. Instead, an implementation needs to be configured with the interface version to use.

1.8 Vendor Extensible Fields

This protocol cannot be extended by any party other than Microsoft.

This protocol uses Win32 error codes as defined in [MS-ERREF] section 2.2. Vendors SHOULD reuse those values with their indicated meaning. Choosing any other value runs the risk of a collision in the future.

1.9 Standards Assignments

The following table lists the universally unique identifier (UUID) value for the **authzr** interface specified in section 3.1.

Parameter	Value	Reference
UUID for authzr	0b1c2170-5732-4e0e-8cd3-d9b16f3b84d7	[C706]

2 Messages

2.1 Transport

This protocol uses the following RPC protocol sequences as specified in [MS-RPCE] (section 2.1.1.1 for TCP/IP - NCACN_IP_TCP and section 2.1.1.2 for SMB - NCACN_NP):

- RPC over TCP/IP

This protocol uses the following RPC endpoints:

- Dynamic endpoints as specified in [C706] part 4

This protocol MUST use the following UUIDs:

- authzr interface: 0b1c2170-5732-4e0e-8cd3-d9b16f3b84d7
- Object UUIDs: 9a81c2bd-a525-471d-a4ed-49907c0b23da and 5fc860e0-6f6e-4fc2-83cd-46324f25e90b

2.2 Common Data Types

This protocol MUST indicate to the RPC runtime that it is to support both the NDR and NDR64 transfer syntaxes and provide a negotiation mechanism for determining which transfer syntax will be used ([MS-RPCE] section 3.1.1.5.1.1).

The following data types are specified in [MS-DTYP]:

Data type name	Section
ACCESS_MASK	2.4.3
BYTE	2.2.6
DWORD	2.2.9
GUID	2.3.4
LARGE_INTEGER	2.3.5
LONG	2.2.27
LONG64	2.2.31
LUID	2.3.7
OBJECT_TYPE_LIST	2.3.9
PVOID	2.2.59
RPC_SID	2.4.2.3
RPC_SID_IDENTIFIER_AUTHORITY	2.4.1.1
SID	2.4.2
SID_IDENTIFIER_AUTHORITY	2.4.1
ULONG	2.2.51
ULONG64	2.2.54

Data type name	Section
USHORT	2.2.58
VOID	2.2.59
WCHAR	2.2.60
WORD	2.2.61

2.2.1 Data Types

This protocol defines the following data type.

DataType name	Section	Description
AUTHZR_HANDLE	2.2.1.1	Represents an explicit RPC binding handle associated with an authzr interface. Used to maintain security information about a principal.

2.2.1.1 AUTHZR_HANDLE

The AUTHZR_HANDLE data type is used to maintain security information about a principal; it represents an explicit RPC binding handle associated with an authzr interface. This is used to identify the client context when calling methods in this protocol, and the server tracks the AUTHZR_HANDLE for each **ClientContext** ADM element in its **ClientContextList**.

```
typedef [context_handle] PVOID AUTHZR_HANDLE;
```

2.2.2 Enumerations

This protocol uses the following enumeration.

Enumeration name	Section	Description
AUTHZ_CONTEXT_INFORMATION_CLASS	2.2.2.1	References the security attributes of a principal represented by an AUTHZR_HANDLE.
AUTHZ_SECURITY_ATTRIBUTE_OPERATION	2.2.2.2	Identifies operation types on a client context object.
AUTHZ_SID_OPERATION	2.2.2.3	Indicates the type of SID operations that can be made by a call to the **AuthzrModifySids** method.

2.2.2.1 AUTHZ_CONTEXT_INFORMATION_CLASS

The AUTHZ_CONTEXT_INFORMATION_CLASS enumeration is used to indicate security attributes of a principal represented by an AUTHZR_HANDLE.

```
typedef enum _AUTHZ_CONTEXT_INFORMATION_CLASS {
    AuthzContextInfoUserSid = 1,
```

```
    AuthzContextInfoGroupsSids = 2,
    AuthzContextInfoRestrictedSids = 3,
    ReservedEnumValue4 = 4,
    ReservedEnumValue5 = 5,
    ReservedEnumValue6 = 6,
    ReservedEnumValue7 = 7,
    ReservedEnumValue8 = 8,
    ReservedEnumValue9 = 9,
    ReservedEnumValue10 = 10,
    ReservedEnumValue11 = 11,
    AuthzContextInfoDeviceSids = 12,
    AuthzContextInfoUserClaims = 13,
    AuthzContextInfoDeviceClaims = 14,
    ReservedEnumValue15 = 15,
    ReservedEnumValue16 = 16
} AUTHZ_CONTEXT_INFORMATION_CLASS;
```

AuthzContextInfoUserSid: Indicates the principal's user SID and its attribute.

AuthzContextInfoGroupsSids: Indicates the groups to which the principal belongs and their attributes.

AuthzContextInfoRestrictedSids: Indicates the restricted SIDs in the principal's security context and their attributes.

ReservedEnumValue4: Not used.

ReservedEnumValue5: Not used.

ReservedEnumValue6: Not used.

ReservedEnumValue7: Not used.

ReservedEnumValue8: Not used.

ReservedEnumValue9: Not used.

ReservedEnumValue10: Not used.

ReservedEnumValue11: Not used.

AuthzContextInfoDeviceSids: Indicates the groups to which the device principal belongs and their attributes.

AuthzContextInfoUserClaims: Indicates the user's security attributes information.

AuthzContextInfoDeviceClaims: Indicates the device's security attributes information.

ReservedEnumValue15: Not used.

ReservedEnumValue16: Not used.

2.2.2.2 AUTHZ_SECURITY_ATTRIBUTE_OPERATION

The AUTHZ_SECURITY_ATTRIBUTE_OPERATION enumeration structure is used with the AuthzrModifyClaims operation (section 3.1.4.6) to identify operation types on a client context object.

```
typedef enum _AUTHZ_SECURITY_ATTRIBUTE_OPERATION {
```

```
    AUTHZ_SECURITY_ATTRIBUTE_OPERATION_NONE = 0,
    AUTHZ_SECURITY_ATTRIBUTE_OPERATION_REPLACE_ALL = 1,
    AUTHZ_SECURITY_ATTRIBUTE_OPERATION_ADD = 2,
    AUTHZ_SECURITY_ATTRIBUTE_OPERATION_DELETE = 3,
    AUTHZ_SECURITY_ATTRIBUTE_OPERATION_REPLACE = 4
} AUTHZ_SECURITY_ATTRIBUTE_OPERATION;
```

AUTHZ_SECURITY_ATTRIBUTE_OPERATION_NONE: No operation will be performed.

AUTHZ_SECURITY_ATTRIBUTE_OPERATION_REPLACE_ALL: The ImpersonationAccessToken on the specified client context will be replaced.

AUTHZ_SECURITY_ATTRIBUTE_OPERATION_ADD: A new claim will be added to the server's ImpersonationAccessToken associated with the specified client context.

AUTHZ_SECURITY_ATTRIBUTE_OPERATION_DELETE: An existing claim will be deleted from the **ImpersonationAccessToken** array associated with the specified client context if it is present in that array.

AUTHZ_SECURITY_ATTRIBUTE_OPERATION_REPLACE: An existing claim will be replaced in the **ImpersonationAccessToken** array associated with the specified client context if it is present in the array.

2.2.2.3 AUTHZ_SID_OPERATION

The AUTHZ_SID_OPERATION enumeration indicates the type of SID operations that can be made by a call to the **AuthzrModifySids** operation (section 3.1.4.7).

```
typedef enum _AUTHZ_SID_OPERATION {
    AUTHZ_SID_OPERATION_NONE = 0,
    AUTHZ_SID_OPERATION_REPLACE_ALL = 1,
    AUTHZ_SID_OPERATION_ADD = 2,
    AUTHZ_SID_OPERATION_DELETE = 3,
    AUTHZ_SID_OPERATION_REPLACE = 4
} AUTHZ_SID_OPERATION;
```

AUTHZ_SID_OPERATION_NONE: Do not modify anything.

AUTHZ_SID_OPERATION_REPLACE_ALL: Replace the existing SIDs with the specified SIDs. If replacement SIDs are not specified, delete the existing SIDs. This operation can be specified only once and must be the only operation specified.

AUTHZ_SID_OPERATION_ADD: Add a new SID. If the SID already exists, fail the call.

AUTHZ_SID_OPERATION_DELETE: Delete the specified SID. If the specified SID is not found, fail the call without taking action.

AUTHZ_SID_OPERATION_REPLACE: Replace the existing SID with the specified SID. If the SID does not exist, add the specified SID.

2.2.3 Structures

This protocol uses the following structures.

Structure name	Section	Description
AUTHZR_ACCESS_REPLY	2.2.3.1	Defines the contents of a remote access check reply.
AUTHZR_ACCESS_REQUEST	2.2.3.2	Defines the contents of a remote access check request.
AUTHZR_CONTEXT_INFORMATION	2.2.3.3	Contains security information about a principal.
AUTHZR_SECURITY_ATTRIBUTE_STRING_VALUE	2.2.3.4	Specifies a string value associated with a security attribute.
AUTHZR_SECURITY_ATTRIBUTE_V1	2.2.3.5	Specifies a security attribute and one or more value pairs.
AUTHZR_SECURITY_ATTRIBUTE_V1_VALUE	2.2.3.6	Specifies a value associated with a security attribute.
AUTHZR_SECURITY_ATTRIBUTES_INFORMATION	2.2.3.7	Specifies one or more security attributes and values.
AUTHZR_SID_AND_ATTRIBUTES	2.2.3.8	Contains information about the security identifiers (SIDs) in a token.
AUTHZR_TOKEN_GROUPS	2.2.3.9	Represents a security identifier (SID) and its attributes.
AUTHZR_TOKEN_USER	2.2.3.10	Identifies the user associated with a token.
SR_SD	2.2.3.11	Contains a self-relative security descriptor.

2.2.3.1 AUTHZR_ACCESS_REPLY

The AUTHZR_ACCESS_REPLY structure defines the contents of a remote access check reply.

```
typedef struct _AUTHZR_ACCESS_REPLY {
    [range(0, 256)] DWORD ResultListLength;
    [size_is(ResultListLength)] ACCESS_MASK* GrantedAccessMask;
    [size_is(ResultListLength)] DWORD* Error;
} AUTHZR_ACCESS_REPLY;
```

ResultListLength: The number of elements in the *GrantedAccessMask* and *Error* arrays. This number matches the number of entries in the object type list structure used in the access check. The length MUST be between 1 and 256. If no object type is used to represent the object, *ResultListLength* MUST be set to 1.

GrantedAccessMask: A pointer to an array of granted access masks.

Error: A pointer to an array of DWORD error code results for each element of the array.

2.2.3.2 AUTHZR_ACCESS_REQUEST

The AUTHZR_ACCESS_REQUEST structure defines the contents of a remote access check request.

```
typedef struct _AUTHZR_ACCESS_REQUEST {
```

```
    ACCESS_MASK DesiredAccess;
    RPC_SID* PrincipalSelfSid;
    [range(0, 256)] DWORD ObjectTypeListLength;
    [size_is(ObjectTypeListLength)] OBJECT_TYPE_LIST* ObjectTypeList;
} AUTHZR_ACCESS_REQUEST;
```

DesiredAccess: The type of access to test.

PrincipalSelfSid: A pointer to the security identifier (SID) to use for the principal self SID in the access control list (ACL).

ObjectTypeListLength: The number of elements in the *ObjectTypeList* array.

ObjectTypeList: A pointer to an array of OBJECT_TYPE_LIST structures in the object tree for the object.

2.2.3.3 AUTHZR_CONTEXT_INFORMATION

The AUTHZR_CONTEXT_INFORMATION structure contains security information about a principal.

```
typedef struct _AUTHZR_CONTEXT_INFORMATION {
    USHORT ValueType;
    [switch_is(ValueType)] union AUTHZR_CONTEXT_INFORMATION_UNION {
        [case(0x1)]
            AUTHZR_TOKEN_USER* pTokenUser;
        [case(0x2, 0x3, 0xC)]
            AUTHZR_TOKEN_GROUPS* pTokenGroups;
        [case(0xD, 0xE)]
            AUTHZR_SECURITY_ATTRIBUTES_INFORMATION* pTokenClaims;
    } ContextInfoUnion;
} AUTHZR_CONTEXT_INFORMATION;
```

ValueType: Identifies the type of the ContextInfoUnion member.

Value	Meaning
0x0001 (user)	ContextInfoUnion contains an AUTHZR_TOKEN_USER structure, as specified in section 2.2.3.10.
0x0002 (groups) 0x0003 (restricted groups) 0x000C (device groups)	ContextInfoUnion contains an AUTHZR_TOKEN_GROUPS structure, as specified in section 2.2.3.9.
0x000D (user claim) 0x000E (device claim)	ContextInfoUnion contains an AUTHZR_SECURITY_ATTRIBUTES_INFORMATION structure, as specified in section 2.2.3.7.

ContextInfoUnion: A pointer to an AUTHZR_TOKEN_USER, AUTHZR_TOKEN_GROUPS, or AUTHZR_SECURITY_ATTRIBUTES_INFORMATION structure, depending on the value of ValueType.

2.2.3.4 AUTHZR_SECURITY_ATTRIBUTE_STRING_VALUE

The AUTHZR_SECURITY_ATTRIBUTE_STRING_VALUE structure contains the string value of a claim.

```
typedef struct _AUTHZR_SECURITY_ATTRIBUTE_STRING_VALUE {
    [range(2, 32768)] ULONG Length;
    [string] [size_is(Length)] WCHAR* Value;
} AUTHZR_SECURITY_ATTRIBUTE_STRING_VALUE;
```

Length: The length of the string in the Value parameter.

Value: A Unicode string containing the pass-through string value of the claim.

2.2.3.5 AUTHZR_SECURITY_ATTRIBUTE_V1

The AUTHZR_SECURITY_ATTRIBUTE_V1 structure specifies one or more security attribute and value pairs that are associated with a remote authorization context.

```
typedef struct _AUTHZR_SECURITY_ATTRIBUTE_V1 {
    [range(2, 256)] ULONG Length;
    [string] [size_is(Length)] WCHAR* Value;
    USHORT ValueType;
    USHORT Reserved;
    ULONG Flags;
    [range(0, 1024)] ULONG ValueCount;
    [size_is(ValueCount)] AUTHZR_SECURITY_ATTRIBUTE_V1_VALUE* Values;
} AUTHZR_SECURITY_ATTRIBUTE_V1;
```

Length: The length of the Value parameter, in bytes. MUST be between 2 and 256.

Value: A Unicode string containing the security value. This string MUST be between 2 and 256 bytes in length, inclusive.

ValueType: A union tag value indicating the type of information contained in **Values** member.

Reserved: Reserved. This member MUST be set to zero when sent and MUST be ignored when received.

Flags: MUST be zero or a combination of one or more of the following values.

Value	Description
AUTHZ_SECURITY_ATTRIBUTE_NON_INHERITABLE 0x00000001	This security attribute is not inherited across processes.
AUTHZ_SECURITY_ATTRIBUTE_VALUE_CASE_SENSITIVE 0x00000002	The value of the attribute is case sensitive. This flag is valid for values that contain string types.

ValueCount: The number of attribute and value pairs pointed to by the **Values** member. The number of attribute and value pairs MUST be between 0 and 1,024, inclusive.

Values: An array of AUTHZR_SECURITY_ATTRIBUTE_V1_VALUE structures, as defined in section 2.2.3.6. Each structure contains a security attribute and value pair.

2.2.3.6 AUTHZR_SECURITY_ATTRIBUTE_V1_VALUE

The AUTHZR_SECURITY_ATTRIBUTE_V1_VALUE structure defines a claim.

```
typedef struct _AUTHZR_SECURITY_ATTRIBUTE_V1_VALUE {
    USHORT ValueType;
    [switch_is(ValueType)] union AUTHZR_SECURITY_ATTRIBUTE_UNION {
        [case(0x1)]
            LONG64 Int64;
        [case(0x2, 0x6)]
            ULONG64 Uint64;
        [case(0x3)]
            AUTHZR_SECURITY_ATTRIBUTE_STRING_VALUE String;
    } AttributeUnion;
} AUTHZR_SECURITY_ATTRIBUTE_V1_VALUE;
```

ValueType: Identifies the type of the AttributeUnion member.

Value	Meaning
0x0001	AttributeUnion contains a LONG64 value.
0x0002, 0x0006	AttributeUnion contains a ULONG64 value.
0x0003	AttributeUnion contains an AUTHZR_SECURITY_ATTRIBUTE_STRING_VALUE structure, as specified in section 2.2.3.4.

AttributeUnion: A LONG64, ULONG64, or AUTHZR_SECURITY_ATTRIBUTE_STRING_VALUE, depending on the value of ValueType.

2.2.3.7 AUTHZR_SECURITY_ATTRIBUTES_INFORMATION

The AUTHZR_SECURITY_ATTRUBUTES_INFORMATION structure specifies one or more security attributes.

```
typedef struct _AUTHZR_SECURITY_ATTRIBUTES_INFORMATION {
    USHORT Version;
    USHORT Reserved;
    [range(0, 1024)] ULONG AttributeCount;
    [size_is(AttributeCount)] AUTHZR_SECURITY_ATTRIBUTE_V1* Attributes;
} AUTHZR_SECURITY_ATTRIBUTES_INFORMATION;
```

Version: The version of this structure. This value MUST be set to 0x0001.

Reserved: Reserved. This member MUST be set to zero when sent and MUST be ignored when received.

AttributeCount: The number of attributes specified by the **Attribute** member. The number of attributes MUST be between zero and 1,024, inclusive.

Attributes: A pointer to an array of AUTHZR_SECURITY_ATTRIBUTE_V1 structures, defined in section 2.2.3.5.

2.2.3.8 AUTHZR_SID_AND_ATTRIBUTES

The AUTHZR_SID_AND_ATTRIBUTES structure contains information about the security identifiers (SIDs) in a token.

```
typedef struct _AUTHZR_SID_AND_ATTRIBUTES {
    RPC_SID* Sid;
    DWORD Attributes;
} AUTHZR_SID_AND_ATTRIBUTES;
```

Sid: A SID structure, as specified in [MS-DTYP] section 2.4.2.3. This is a pass-through value and SHOULD NOT be interpreted by the RAZA protocol.

Attributes: Specifies attributes associated with the SID. This is a pass-through value and SHOULD NOT be interpreted by the RAZA protocol.

2.2.3.9 AUTHZR_TOKEN_GROUPS

The AUTHZR_TOKEN_GROUPS structure represents a security identifier (SID) and its attributes.

```
typedef struct _AUTHZR_TOKEN_GROUPS {
    DWORD GroupCount;
    [size_is(GroupCount)] AUTHZR_SID_AND_ATTRIBUTES Groups[];
} AUTHZR_TOKEN_GROUPS;
```

GroupCount: Indicates the number of structures in the Groups array.

Groups: An array of AUTHZR_SID_AND_ATTRIBUTES structures (section 2.2.3.8) representing groups associated with the token.

2.2.3.10 AUTHZR_TOKEN_USER

The AUTHZR_TOKEN_USER structure identifies the user associated with a token.

```
typedef struct _AUTHZR_TOKEN_USER {
    AUTHZR_SID_AND_ATTRIBUTES User;
} AUTHZR_TOKEN_USER;
```

User: Contains an AUTHZR_SID_AND_ATTRIBUTES structure (section 2.2.3.8) representing the user associated with the access token.

2.2.3.11 SR_SD

The SR_SD structure defines a self-relative security descriptor. A self-relative security descriptor contains the security descriptor structure itself and the necessary security information associated with the descriptor.

```
typedef struct _SR_SD {
    [range(20, 131228)] DWORD dwLength;
    [size_is(dwLength)] BYTE* pSrSd;
```

```
} SR_SD;
```

dwLength: The length, in bytes, of the data pointed to in the **pSrSd** member.

pSrSd: A pointer to a self-relative security descriptor.

3 Protocol Details

The Remote Authorization Protocol is used to approximate an access control decision that would be made when a given principal attempts to access a hypothetical resource on a remote service that is protected with a given authorization policy.

All remote authorization methods return 0x00000000 on success; otherwise, they return a 32-bit, nonzero Win32 error code. For more details about Win32 error values, see [MS-ERREF].

Unless otherwise specified, the pointer type for the RAZA RPC interface is pointer_default(unique). Method calls are received at a dynamically assigned endpoint ([MS-RPCE] section 2.1.1.1). The endpoints for the Netlogon service are negotiated by the RPC endpoint mapper ([MS-RPCE] section 2.1.1.1).

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

3.1 authzr Server Details

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The organization is provided to explain 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 specified in this document.

The RAZA server maintains one or more of the following abstract data types as abstract variables:

- **ImpersonationAccessToken (Public):** A Token/Authorization Context (see [MS-DTYP] section 2.5.2).
- **ClientContext:**A data structure containing the following members:
  - **RPCClient:** An AUTHZR_HANDLE structure (section 2.2.1.1).
  - **AuthzContext:** An **ImpersonationAccessToken**.

Additionally, the RAZA server MUST maintain the following data structure:

- **ClientContextList:** A list of **ClientContext** objects.

3.1.2 Timers

None.

3.1.3 Initialization

The authzr server registers an endpoint with RPC over TCP/IP. The authzr server MUST register the Negotiate security support provider authentication_type constant [0x09] as the security provider ([MS-RPCE] section 3.3.3.3.1.3) used by the RPC interface.

The **ClientContextList** is emptied during initialization.

3.1.4 Message Processing Events and Sequencing Rules

This interface includes the following methods.

Method	Description
AuthzrFreeContext	Opnum: 0
AuthzrInitializeContextFromSid	Opnum: 1
AuthzrInitializeCompoundContext	Opnum: 2
AuthzrAccessCheck	Opnum: 3
AuthzGetInformationFromContext	Opnum: 4
AuthzrModifyClaims	Opnum: 5
AuthzrModifySids	Opnum: 6

All methods MUST NOT throw exceptions.

3.1.4.1 AuthzrFreeContext (Opnum 0)

The AuthzrFreeContext method (opnum 0) frees all remote structures and memory associated with the client context identified by the *ContextHandle* parameter.

```
DWORD AuthzrFreeContext(
        [in, out] AUTHZR_HANDLE* ContextHandle);
```

ContextHandle: A pointer to an AUTHZR_HANDLE structure, as defined in section 2.2.1.1. This handle indicates the client context to be freed.

Return Values:

If the function succeeds, it MUST return 0x00000000.

If the function fails, it MUST return a nonzero 32-bit error code.

When a remote authorization server receives this message, it MUST look up the **ClientContext** structure in the **ClientContextTable** ADM element and free all structures and memory associated with the **ClientContext**.

3.1.4.2 AuthzrInitializeContextFromSid (Opnum 1)

The AuthzrInitializeContextFromSid method (opnum 1) creates a client context from a given security identifier (SID). For domain SIDs, token group and claim attributes will be retrieved from Active Directory through Kerberos.

```
DWORD AuthzrInitializeContextFromSid(
        [in] handle_t Binding,
        [in] DWORD Flags,
        [in] RPC_SID* Sid,
        [in] [unique] LARGE_INTEGER* pExpirationTime,
        [in] LUID Identifier,
        [out] AUTHZR_HANDLE* ContextHandle);
```

Binding: A primitive RPC handle that identifies a particular client/server binding.

Flags: Indicates the type of logon behavior when initializing the client context. The following flags are defined.

Value	Description
0x00000000	When no flags are set, AuthzInitializeContextFromSid attempts to retrieve the user's token group information by performing an S4U logon.
AUTHZ_COMPUTE_PRIVILEGES (0x00000008)	AuthzInitializeContextFromSid retrieves privileges for the new context. If this function performs an S4U logon, it retrieves privileges from the token. Otherwise, it retrieves privileges from all SIDs in the context.

All other bits MUST be set to zero.

Sid: A pointer to the SID of the principal for whom a remote client context will be created. This MUST be a valid user or computer account.

pExpirationTime: Reserved. This parameter MUST be set to NULL when sent and MUST be ignored when received.

Identifier: Reserved. This parameter MUST be set to zero when sent and MUST be ignored when received.

ContextHandle: A pointer to an AUTHZR_HANDLE structure, as defined in section 2.2.1.1.

Return Values:

If the function succeeds, the function MUST return 0x00000000.

If the function fails, it MUST return a nonzero error code.

When a RAZA server receives this message, the server MUST perform the following:

1. If any bits other than 0x00000008 are set in *Flags*, the server MUST return ERROR_INVALID_PARAMETER.
2. Call *LsarOpenPolicy* ([MS-LSAT] section 3.1.4.2) with the following as input:
   - *SystemName*: NULL.
   - *DesiredAccess*: Contains the bit value 0x00000800 for POLICY_LOOKUP_NAMES.
3. Call *LsarLookupSids* ([MS-LSAT] section 3.1.4.11) on the returned **PolicyHandle**.
   - *PolicyHandle*: The **PolicyHandle** returned from the aforementioned *LsarOpenPolicy*.
   - *SidEnumBuffer*: The **SidInfo** part of this structure contains the *Sid* parameter. The **Entries** part of this structure is set to 1. **LookupLevel** is set to *LsapLookupWksta*.

   The return values from LsarLookupSids are as follows:

   - **ReferencedDomains list**: The domain name is found as follows:
     1. Locate the entry in the **TranslatedNames** list that corresponds to the SID in question. This entry contains a **Names** structure with a **DomainIndex**.

2. Find the **ReferencedDomains** list entry with an index that matches the **DomainIndex** from the structure in the preceding step. The domain name is found in the **Name** field of the **Domains** structure.

- **TranslatedNames**: Contains the **UserName** in the **Name** field of the **Names** structure of the entry in the list corresponding to the SID in question (from the *SidEnumBuffer* input list).

4. Perform a Kerberos S4U2Self service ticket request using the S4U2self KRB_TGS_REQ/KRB_TGS_REP protocol extension as specified in [MS-SFU] section 3.1.5.1.1.1.
   - The *userName* MUST be set to the user name obtained in step 2.
   - The *userRealm* MUST be set to the domain name of the obtained in step 2.
   - The *chksum* MUST be set as specified in [MS-SFU] section 2.2.2.
   - The *auth-package* MUST be set to "Kerberos".
5. Initialize and populate an **ImpersonationAccessToken** as specified in [MS-KILE] section 3.4.5.3.
6. Allocate and initialize a new AUTHZR_HANDLE structure, as defined in section 2.2.1.1, and assign **ContextHandle** to the new structure.
7. Allocate memory for a new **ClientContext** object, set the **RPCClient** member to the AUTHZR_HANDLE initialized in step 6, and set **AuthzContext** to the **ImpersonationAccessToken** initialized in step 5.
8. Append the **ClientContext** object created in step 7 to the **ClientContextList**.

3.1.4.3 AuthzrInitializeCompoundContext (Opnum 2)

The AuthzrInitializeCompoundContext method (opnum 2) creates a compound context from two specified context handles.

```
DWORD AuthzrInitializeCompoundContext(
        [in] AUTHZR_HANDLE UserContextHandle,
        [in] AUTHZR_HANDLE DeviceContextHandle,
        [out] AUTHZR_HANDLE* CompoundContextHandle);
```

UserContextHandle: An AUTHZR_HANDLE structure, as defined in section 2.2.1.1, that represents the user context for the compound context.

DeviceContextHandle: An AUTHZR_HANDLE structure, as defined in section 2.2.1.1, that represents the device context for the compound context.

CompoundContextHandle: A pointer to an AUTHZR_HANDLE structure, as defined in section 2.2.1.1.

Return Values:

If the function succeeds, the function MUST return 0x00000000. If the function fails, it MUST return a nonzero value.

When a RAZA server receives this message, the server MUST perform the following:

1. Allocate a new **ImpersonationAccessToken**.

2. Copy the **ImpersonationAccessToken.Sids** array in the **ImpersonationAccessToken** of the **UserContextHandle** into the **ImpersonationAccessToken.Sids** array in the **ImpersonationAccessToken** created in step 1.

3. Copy the **ImpersonationAccessToken.UserIndex** field in the **ImpersonationAccessToken** of the **UserContextHandle** to the **ImpersonationAccessToken.UserIndex** field in the **ImpersonationAccessToken** created in step 1.

4. Copy the **ImpersonationAccessToken.UserClaims** array in the **ImpersonationAccessToken** of the **UserContextHandle** to the **ImpersonationAccessToken.UserClaims** array in the **ImpersonationAccessToken** created in step 1.

5. Copy the **ImpersonationAccessToken.Sids** array in the **ImpersonationAccessToken** of the *DeviceContextHandle* into the **ImpersonationAccessToken.DeviceSids** array in the **ImpersonationAccessToken** created in step 1.

6. Copy the **ImpersonationAccessToken.UserIndex** field in the **ImpersonationAccessToken** of the *DeviceContextHandle* to the **ImpersonationAccessToken.DeviceIndex** field in the **ImpersonationAccessToken** created in step 1.

7. Copy the **ImpersonationAccessToken.UserClaims** array in the **ImpersonationAccessToken** of the *DeviceContextHandle* to the **ImpersonationAccessToken.DeviceClaims** array in the **ImpersonationAccessToken** created in step 1.

8. Allocate and initialize a new AUTHZR_HANDLE structure.

9. Allocate a new **ClientContext** object, set the **RPCClient** member to the AUTHZR_HANDLE allocated in step 8, and set the **AuthzContext** member to the **ImpersonationAccessToken** created in step 1.

10. Add the new **ClientContext** object to the **ClientContextList.**

3.1.4.4 AuthzrAccessCheck (Opnum 3)

The AuthzrAccessCheck method (opnum 3) determines which access bits can be granted to a client for a given set of security descriptors. The AUTHZR_ACCESS_REPLY structure returns an array of granted access masks and error status.

```
DWORD AuthzrAccessCheck(
        [in] AUTHZR_HANDLE ContextHandle,
        [in] DWORD Flags,
        [in] AUTHZR_ACCESS_REQUEST* pRequest,
        [in] [range(1, 16)] DWORD SecurityDescriptorCount,
        [in] [size_is(SecurityDescriptorCount)] SR_SD* pSecurityDescriptors,
        [in, out] AUTHZR_ACCESS_REPLY* pReply);
```

ContextHandle: An AUTHZR_HANDLE structure, as defined in section 2.2.1.1, containing the client context handle.

Flags: Reserved. This parameter MUST be set to zero.

pRequest: A pointer to an AUTHZR_ACCESS_REQUEST structure, as defined in section 2.2.3.2. This structure contains the body of the "what-if" access check request.

SecurityDescriptorCount: The number of security descriptors in the *pSecurityDescriptors* parameter, not including the primary security descriptor.

pSecurityDescriptors: A pointer to an array of SR_SD structures, as defined in section 2.2.3.11. The first entry in this array is the primary security descriptor, and it will be used as the security descriptor for the AccessCheck evaluation.

pReply: A pointer to an AUTHZR_ACCESS_REPLY structure, as defined in section 2.2.3.1. This parameter will contain the body of the access check response.

Return Values:

If the function succeeds, the function MUST return 0x00000000.

If the function fails, it MUST return a nonzero error code.

When a RAZA server receives this message, the server MUST perform the following:

- Check that the upper 16 bits of the **Flags** parameter are set to zero, and if not, return a nonzero error code.
- If the client connects to the server using ObjectUUID as 5fc860e0-6f6e-4fc2-83cd-46324f25e90b then remove all ACEs of type SYSTEM_SCOPED_POLICY_ID_ACE ([MS-DTYP] section 2.4.4.16) from the pSecurityDescriptors[0] parameter.
- Perform an AccessCheck evaluation using the algorithm specified in [MS-DTYP] section 2.5.3.2, where the preceding parameters are mapped to the parameter names of the algorithm described according to the following table.

AccessCheck pseudocode parameter	RAZA protocol AuthzrAccessCheck evaluation
SecurityDescriptor	*pSecurityDescriptors[0]*
Token (Authorization Context)	The **ImpersonationAccessToken** in the **ClientContext** object associated with the *ContextHandle*
Access Request mask	The **DesiredAccess** member of the AUTHZR_ACCESS_REQUEST structure pointed to by *pRequest*
Object Tree	The **ObjectTypeList** member of the AUTHZR_ACCESS_REQUEST structure pointed to by *pRequest*
PrincipalSelfSubst SID	The **PrincipalSelfSid** member of the AUTHZR_ACCESS_REQUEST structure pointed to by *pRequest*
GrantedAccess	The memory location of the **GrantedAccessMask** member of the AUTHZR_ACCESS_REPLY structure pointed to by *pRequest*

3.1.4.5 AuthzGetInformationFromContext (Opnum 4)

The AuthzGetInformationFromContext method (opnum 4) returns information about the identified client context.

```
DWORD AuthzGetInformationFromContext(
        [in] AUTHZR_HANDLE ContextHandle,
        [in] AUTHZ_CONTEXT_INFORMATION_CLASS InfoClass,
        [out] AUTHZR_CONTEXT_INFORMATION** ppContextInformation);
```

ContextHandle: An AUTHZR_HANDLE structure, as defined in section 2.2.1.1. Represents the client context to retrieve information from.

InfoClass: An AUTHZ_CONTEXT_INFORMATION_CLASS enumeration, as defined in section 2.2.2.1. Possible values for this field are specified in section 2.2.2.1.

ppContextInformation: A two-layer pointer to an AUTHZR_CONTEXT_INFORMATION structure, as defined in section 2.2.3.3. Used to return the context information.

Return Values:

If the function succeeds, the function MUST return 0x00000000.

If the function fails, it MUST return a nonzero error code value.

When a RAZA server receives this message, the server MUST perform the following:

- If the *InfoClass* parameter is one of the following values, the RAZA server MUST:

  1. Initialize a new AUTHZR_CONTEXT_INFORMATION structure.

  2. Set the *ppContextInformation* parameter to the memory address of the new structure.

  3. Perform the corresponding action using the **ImpersonationAccessToken** in the **ClientContext** object identified in the *ContextHandle* parameter:

     - **AuthzContextInfoUserSid** (1):

       1. Set the **ValueType** member in the new AUTHZR_CONTEXT_INFORMATION object to 1.

       2. Set the **pTokenUser** member of the new AUTHZR_CONTEXT_INFORMATION object to the address of the element at the **ImpersonationAccessToken.UserIndex** of the **ImpersonationAccessToken.Sids** array.

     - **AuthzContextInfoGroupsSids** (2):

       1. Set the **ValueType** member in the new AUTHZR_CONTEXT_INFORMATION object to 2.

       2. Set the **pTokenGroups** member of the new AUTHZR_CONTEXT_INFORMATION object to the value of the **ImpersonationAccessToken.Sids** member.

     - **AuthzContextInfoRestrictedSids** (3):

       1. Set the **ValueType** member in the new AUTHZR_CONTEXT_INFORMATION object to 3.

       2. Set the **pTokenGroups** member of the new AUTHZR_CONTEXT_INFORMATION object to the value of the **ImpersonationAccessToken.RestrictedSids** member.

     - **AuthzContextInfoDeviceSids** (12):

       1. Set the **ValueType** member in the new AUTHZR_CONTEXT_INFORMATION object to 12.

       2. Set the **pTokenGroups** member of the new AUTHZR_CONTEXT_INFORMATION object to the value of the **ImpersonationAccessToken.DeviceSids** member.

     - **AuthzContextInfoUserClaims** (13):

1. Set the **ValueType** member in the new AUTHZR_CONTEXT_INFORMATION object to 13.

2. Set the **pTokenClaims** member of the new AUTHZR_CONTEXT_INFORMATION object to the value of the **ImpersonationAccessToken.Claims** member.

- **AuthzContextInfoDeviceClaims** (14):

1. Set the **ValueType** member in the new AUTHZR_CONTEXT_INFORMATION object to 14.

2. Set the **pTokenClaims** member of the new AUTHZR_CONTEXT_INFORMATION object to the value of the **ImpersonationAccessToken.DeviceClaims** member.

- If the *InfoClass* parameter is any other value, the requested information is not supported. The RAZA server MUST set *ppContextInformation* to NULL and return a nonzero error code value.

3.1.4.6 AuthzrModifyClaims (Opnum 5)

The AuthzrModifyClaims method (opnum 5) modifies information about the identified client context.

```
DWORD AuthzrModifyClaims(
        [in] AUTHZR_HANDLE ContextHandle,
        [in] AUTHZ_CONTEXT_INFORMATION_CLASS ClaimClass,
        [in] [range(1, 65535)] DWORD OperationCount,
        [in] [size_is(OperationCount)] AUTHZ_SECURITY_ATTRIBUTE_OPERATION* pClaimOperations,
        [in] [unique] AUTHZR_SECURITY_ATTRIBUTES_INFORMATION* pClaims);
```

ContextHandle: An AUTHZR_HANDLE structure, as defined in section 2.2.1.1. Represents the client context to modify.

ClaimClass: An AUTHZ_CONTEXT_INFORMATION_CLASS enumeration, as defined in section 2.2.2.1. Indicates the claim class.

OperationCount: The number of operations to be performed.

pClaimOperations: A pointer to an array of AUTHZ_SECURITY_ATTRIBUTE_OPERATION enumerations, as defined in section 2.2.2.2. Specifies the operations to be performed on each claim.

pClaims: A pointer to an array of AUTHZR_SECURITY_ATTRIBUTES_INFORMATION structures, as defined in section 2.2.3.7. Contains the claim(s) used to modify the client context.

Return Values:

If the function succeeds, the function MUST return 0x00000000.

If the function fails, it MUST return a nonzero error code value.

When a RAZA server receives this message, the server MUST perform the following:

If the *InfoClass* parameter is any value other than AuthzContextInfoUserClaims (13) or AuthzContextInfoDeviceClaims (14), the requested modification is not supported. When this happens, the RAZA server MUST set *ppContextInformation* to NULL and return FALSE.

The RAZA server MUST check the first element in the *pClaimOperations* array as indicated by *OperationCount* and perform operations as follows:

- If the value pointed to by the *pClaimOperations* parameter is **AUTHZ_SECURITY_ATTRIBUTE_OPERATION_NONE,** the RAZA server MUST return 0x00000000.

- If the value pointed to by the *pClaimsOperations* parameter is **AUTHZ_SECURITY_ATTRIBUTE_OPERATION_REPLACE_ALL,** the RAZA server MUST perform the following on the **ImpersonationAccessToken** in the **ClientContext** identified by the *ContextHandle*:

    1. Map the *InfoClass* parameter value to the corresponding **ImpersonationAccessToken** array according to the following table and replace it with the *pClaims* **Attribute** member array.

    2. Return 0x00000000 if the operation was a success; otherwise, return a nonzero error code.

- If the value pointed to by the *pClaimOperations* parameter is any other value of AUTHZ_SECURITY_ATTRIBUTE_OPERATION, perform the steps prescribed after the next paragraph.

The RAZA server MUST perform the following steps on each element in the *pClaimOperations* array as indicated by *OperationCount*:

1. If the element is not the first element and the value is **AUTHZ_SECURITY_ATTRIBUTE_OPERATION_NONE** or **AUTHZ_SECURITY_ATTRIBUTE_OPERATION_REPLACE_ALL,** the RAZA server MUST return a nonzero error code. The case in which the first element is one of these values is described above.

2. If the element is **AUTHZ_SECURITY_ATTRIBUTE_OPERATION_ADD,** append the corresponding element in the *pClaims* **Attributes** array to the **ImpersonationAccessToken** array identified according to the following table.

3. If the element is **AUTHZ_SECURITY_ATTRIBUTE_OPERATION_DELETE**, search the **ImpersonationAccessToken** array identified according to the following table for a member whose **Value** member equals the **Value** member of the corresponding element of the **Attributes** array in the *pClaims* parameter. If one is found, delete that element from the identified **ImpersonationAccessToken** array and free any memory associated with that element.

4. If the element is **AUTHZ_SECURITY_ATTRIBUTE_OPERATION_REPLACE,** search the **ImpersonationAccessToken** array identified according to the following table for a member whose **Value** member equals the **Value** member of the corresponding element of the **Attributes** array in the *pClaims* parameter.

    1. If the attribute is located, replace the **Values** member of the located attribute with the replacement **Values** member, which is located in the corresponding **Attributes** array element in the **Attributes** member of *pClaims*. If the replacement **Values** member does not exist, the located attribute is deleted.

    2. If the attribute is not located, it is added using the replacement **Values** member, which is located in the corresponding **Attributes** array element in the **Attributes** member of *pClaims*. If the replacement **Values** member does not exist, the operation is ignored and no failure is reported.

ClaimClass parameter value	Corresponding ImpersonationAccessToken array
AuthzContextInfoUserClaims	**ImpersonationAccessToken.Claims**
AuthzContextInfoDeviceClaims	**ImpersonationAccessToken.DeviceClaims**

3.1.4.7 AuthzrModifySids (Opnum 6)

The AuthzrModifySids method (opnum 6) modifies the list of SIDs associated with the identified client context.

```
DWORD AuthzrModifySids(
        [in] AUTHZR_HANDLE ContextHandle,
        [in] AUTHZ_CONTEXT_INFORMATION_CLASS SidClass,
        [in] [range(1, 65535)] DWORD OperationCount,
        [in] [size_is(OperationCount)] AUTHZ_SID_OPERATION* pSidOperations,
        [in] [unique] AUTHZR_TOKEN_GROUPS* pSids);
```

ContextHandle: An AUTHZR_HANDLE structure, as defined in section 2.2.1.1, representing the client context to be modified.

SidClass: An AUTHZ_CONTEXT_INFORMATION_CLASS enumeration value, as defined in section 2.2.2.1, indicating the SID class.

OperationCount: The number of operations to be performed.

pSidOperations: A pointer to an array of AUTHZ_SID_OPERATION enumeration values that specify the group modifications to be made.

pSids: A pointer to an AUTHZR_TOKEN_GROUPS structure, as defined in section 2.2.3.9, specifying the groups to be modified.

Return Values:

If the function succeeds, it MUST return 0x00000000.

If the function fails, it MUST return a nonzero error code value.

On receipt of this message, a RAZA server MUST complete the following process:

1. If the InfoClass parameter contains any value other than AuthzContextInfoGroupSids (2) or AuthzContextInfoDeviceSids (12), or if the requested modification is not supported, the RAZA server MUST return ERROR_INVALID_PARAMETER.

2. The RAZA server MUST check the first element in the pSidOperations array as indicated by OperationCount.

3. If the value pointed to by the pSidOperations parameter is AUTHZ_SID_OPERATION_NONE, the RAZA server must return 0x00000000.

4. If the value pointed to by the pSidOperations parameter is AUTHZ_SID_OPERATION_REPLACE_ALL, the RAZA server MUST perform the following operations on the ImpersonationAccessToken in the ClientContext identified by the ContextHandle:

    1. Map the InfoClass parameter value to the corresponding ImpersonationAccessToken array according to the following table, and replace it with the pSids Groups member array.

    2. Return 0x00000000 if the operation is successful; otherwise, return a nonzero error code.

5. If the value pointed to by the pSidOperations parameter is any other value of the **AUTHZ_SID_OPERATION** enumeration, continue with the following process.

The RAZA server MUST complete the following process for each element in the pSidOperations array as indicated by OperationCount:

1. If the element is not the first element and the value is AUTHZ_SID_OPERATION_NONE or AUTHZ_SID_OPERATION_REPLACE_ALL, the RAZA server MUST return a nonzero error code. The case in which the first element is one of these values is described earlier.

2. If the element is **AUTHZ_SID_OPERATION_ADD**, search the **ImpersonationAccessToken** array identified according to the following table for a member whose member equals the member of the corresponding element of the **Groups** array in the *pSids* parameter. If one is found, the RAZA server MUST return ERROR_GROUP_EXISTS, otherwise, append the corresponding element in the **Groups** array in the *pSids* parameter to the **ImpersonationAccessToken** array identified according to the following table. If the corresponding element of the **Groups** array in the *pSids* parameter does not exist, then RAZA server MUST fail with ERROR_INVALID_PARAMETER.

3. If the element is AUTHZ_SID_OPERATION_DELETE, search the **ImpersonationAccessToken** array that is identified according to the following table for a member whose member equals the member of the corresponding element of the Groups array in the pSids parameter. If one is found, delete that element from the identified **ImpersonationAccessToken** array and free any memory that was associated with that element. If the search fails, the RAZA server MUST return ERROR_NOT_FOUND. If the corresponding element of the **Groups** array in the *pSids* parameter does not exist, then the RAZA server MUST fail with ERROR_INVALID_PARAMETER.

4. If the element is AUTHZ_SID_OPERATION_REPLACE, search the ImpersonationAccessToken array identified according to the following table for a member whose member equals the member of the corresponding element of the Groups array in the pSids parameter.

    1. If the Sid is located, replace the member of the Sid located in the array with the replacement Sid, which is located in the corresponding Groups array element in the Groups member of the pSids parameter. If the corresponding replacement **Groups** member does not exist, then the RAZA server MUST fail with ERROR_INVALID_PARAMETER.

    2. If the Sid is not located, it is added using the replacement Sid, which is located in the corresponding Groups array element in the Groups member of the pSids parameter. If the corresponding replacement **Groups** member does not exist, then the RAZA server MUST fail with ERROR_INVALID_PARAMETER.

SIDClass parameter value	Corresponding ImpersonationAccessToken array
AuthzContextInfoGroupSids	ImpersonationAccessToken.Sids
AuthzContextInfoDeviceSids	ImpersonationAccessToken.DeviceSids

3.1.5 Timer Events

None.

3.1.6 Other Local Events

None.

4 Protocol Examples

The following example shows a sample call sequence from a client to a server for a typical use of the RAZA protocol to query the permissions available to a given user with the SID S-1-5-21-3448151421-356457007-600757626-4138921 for a resource protected on the server with a security descriptor where built-in administrators and local system have file all access, everyone has read and execute, and the user has read, write, and execute.

- Client sends AuthzrInitializeContextFromSid.

Parameter field	Parameter value
Handle_t	[RPC handle to server]
Flags	0x8
SID	S-1-5-21-3448151421-356457007-600757626-4138921
pExpirationTime	NULL
LUID	{0xdead,0xbeef}

- Client receives AuthzrInitializeContextFromSid.

Parameter field	Parameter value
Status	0
ContextHandle	[context handle](This data is opaque to the client.)

- Client sends AuthzrAccessCheck.

Parameter field	Parameter value
Handle_t	[RPC handle to server]
ContextHandle	[ContextHandle received from server using AuthzrInitializeContextFromSid]

Parameter field	Parameter value
Flags	0x0
*pRequest.***DesiredAccess**	0x02000000 (MAXIMUM_ALLOWED)
*pRequest.***PrincipalSelfSid**	NULL
*pRequest.***ObjectTypeListLength**	0
*pRequest.***ObjectTypeList**	NULL
SecurityDescriptorCount	1
pSecurityDescriptors	01 00 04 80 14 00 00 00 24 00 00 00 00 00 00 00 30 00 00 00 01 02 00 00 00 00 00 05 20 00 00 00 20 02 00 00 01 01 00 00 00 00 00 05 12 00 00 00 02 00 6C 00 04 00 00 00 00 00 18 00 FF 01 1F 00 01 02 00 00 00 00 00 05 20 00 00 00 20 02 00 00 00 00 14 00 FF 01 1F 00 01 01 00 00 00 00 00 05 12 00 00 00 00 00 14 00 A9 00 12 00 01 01 00 00 00 00 00 01 00 00 00 00 00 00 24 00 BF 01 12 00 01 05 00 00 00 00 00 05 15 00 00 00 7D 9D 86 CD 2F 1A 3F 15 7A D5 CE 23 A9 27 3F 00 This is equivalent to the following string value: O:BAG:SYD:(A;;FA;;;BA)(A;;FA;;;SY)(A;;FRFX;;;WD)(A;;FWFRFX;;;S-1-5-21-3448151421-356457007-600757626-4138921)

- Client receives AuthzrAccessCheck.

Parameter field	Parameter value
Status	0
pReply.**ResultListLength**	1
pReply.**GrantedAccessMask**	0x1201BF

Parameter field	Parameter value
	(FILE_GENERIC_READ | FILE_GENERIC_WRITE | FILE_GENERIC_EXECUTE)
pReply.**Error**	ERROR_SUCCESS

- Client sends AuthzrFreeContext.

Parameter field	Parameter value
Handle_t	[RPC handle to server]
ContextHandle	[ContextHandle received from server using AuthzrInitializeContextFromSid]

- Client receives AuthzrFreeContext.

Parameter field	Parameter value
Status	0
ContextHandle	NULL

5 Security

5.1 Security Considerations for Implementers

Use of the RAZA protocol requires the client user to have access to read the user and claim information of security principals that the client is preforming authorization queries on.<2> It is recommended that access to the RAZA interface be limited to a subset of the principals who have access to read account information.

5.2 Index of Security Parameters

None.

6 Appendix A: Full IDL

For ease of implementation, the full IDL is provided below, where "ms-dtyp.idl" refers to the IDL found in [MS-DTYP] Appendix A. The syntax uses the IDL syntax extensions defined in [MS-RPCE] sections 2.2.4 and 3.1.1.5.1. For example, as noted in [MS-RPCE] section 2.2.4.9, a pointer_default declaration is not required and pointer_default(unique) is assumed.

```
import "ms-dtyp.idl";

[uuid(0b1c2170-5732-4e0e-8cd3-d9b16f3b84d7)]
[version(0.0)]
[pointer_default(ptr)]
[ms_union]
interface authzr {
    typedef [context_handle] PVOID AUTHZR_HANDLE;
    typedef struct _AUTHZR_ACCESS_REQUEST {
        ACCESS_MASK DesiredAccess;
        RPC_SID * PrincipalSelfSid;
        [range(0,256)] DWORD ObjectTypeListLength;
        [size_is(ObjectTypeListLength)] OBJECT_TYPE_LIST * ObjectTypeList;
    } AUTHZR_ACCESS_REQUEST;

    typedef struct _SR_SD {
        [range(20,131228)] DWORD dwLength;
        [size_is(dwLength)] BYTE * pSrSd;
    } SR_SD;

    typedef struct _AUTHZR_ACCESS_REPLY {
        [range(0,256)] DWORD ResultListLength;
        [size_is(ResultListLength)] ACCESS_MASK * GrantedAccessMask;
        [size_is(ResultListLength)] DWORD * Error;
    } AUTHZR_ACCESS_REPLY;

    typedef enum _AUTHZ_CONTEXT_INFORMATION_CLASS {
        AuthzContextInfoUserSid = 1,
        AuthzContextInfoGroupsSids = 2,
        AuthzContextInfoRestrictedSids = 3,
        ReservedEnumValue4 = 4,
        ReservedEnumValue5 = 5,
        ReservedEnumValue6 = 6,
        ReservedEnumValue7 = 7,
        ReservedEnumValue8 = 8,
        ReservedEnumValue9 = 9,
        ReservedEnumValue10 = 10,
        ReservedEnumValue11 = 11,
        AuthzContextInfoDeviceSids = 12,
        AuthzContextInfoUserClaims = 13,
        AuthzContextInfoDeviceClaims = 14,
        ReservedEnumValue15 = 15,
        ReservedEnumValue16 = 16
    } AUTHZ_CONTEXT_INFORMATION_CLASS;

    typedef struct _AUTHZR_SID_AND_ATTRIBUTES {
        RPC_SID * Sid;
        DWORD Attributes;
    } AUTHZR_SID_AND_ATTRIBUTES;

    typedef struct _AUTHZR_TOKEN_USER {
        AUTHZR_SID_AND_ATTRIBUTES User;
    } AUTHZR_TOKEN_USER;

    typedef struct _AUTHZR_TOKEN_GROUPS {
        DWORD GroupCount;
        [size_is(GroupCount)] AUTHZR_SID_AND_ATTRIBUTES Groups[];
    } AUTHZR_TOKEN_GROUPS;

    typedef struct _AUTHZR_SECURITY_ATTRIBUTE_STRING_VALUE {
```

```
    [range(2,32768)] ULONG Length;
    [string] [size_is(Length)] WCHAR * Value;
} AUTHZR_SECURITY_ATTRIBUTE_STRING_VALUE;

typedef struct _AUTHZR_SECURITY_ATTRIBUTE_V1_VALUE {
    USHORT ValueType;
    [switch_is(ValueType)] union AUTHZR_SECURITY_ATTRIBUTE_UNION {
        [case(0x1)]
            LONG64 Int64;
        [case(0x2, 0x6)]
            ULONG64 Uint64;
        [case(0x3)]
            AUTHZR_SECURITY_ATTRIBUTE_STRING_VALUE String;
    } AttributeUnion;
} AUTHZR_SECURITY_ATTRIBUTE_V1_VALUE;

typedef struct _AUTHZR_SECURITY_ATTRIBUTE_V1 {
    [range(2,256)] ULONG Length;
    [string] [size_is(Length)] WCHAR * Value;
    USHORT ValueType;
    USHORT Reserved;
    ULONG Flags;
    [range(0,1024)] ULONG ValueCount;
    [size_is(ValueCount)] AUTHZR_SECURITY_ATTRIBUTE_V1_VALUE * Values;
} AUTHZR_SECURITY_ATTRIBUTE_V1;

typedef struct _AUTHZR_SECURITY_ATTRIBUTES_INFORMATION {
    USHORT Version;
    USHORT Reserved;
    [range(0,1024)] ULONG AttributeCount;
    [size_is(AttributeCount)] AUTHZR_SECURITY_ATTRIBUTE_V1 * Attributes;
} AUTHZR_SECURITY_ATTRIBUTES_INFORMATION;

typedef struct _AUTHZR_CONTEXT_INFORMATION {
    USHORT ValueType;
    [switch_is(ValueType)] union AUTHZR_CONTEXT_INFORMATION_UNION {
        [case(0x1)]
            AUTHZR_TOKEN_USER * pTokenUser;
        [case(0x2, 0x3, 0xC)]
            AUTHZR_TOKEN_GROUPS * pTokenGroups;
        [case(0xD, 0xE)]
            AUTHZR_SECURITY_ATTRIBUTES_INFORMATION * pTokenClaims;
    } ContextInfoUnion;
} AUTHZR_CONTEXT_INFORMATION;

typedef enum _AUTHZ_SECURITY_ATTRIBUTE_OPERATION {
    AUTHZ_SECURITY_ATTRIBUTE_OPERATION_NONE = 0,
    AUTHZ_SECURITY_ATTRIBUTE_OPERATION_REPLACE_ALL = 1,
    AUTHZ_SECURITY_ATTRIBUTE_OPERATION_ADD = 2,
    AUTHZ_SECURITY_ATTRIBUTE_OPERATION_DELETE = 3,
    AUTHZ_SECURITY_ATTRIBUTE_OPERATION_REPLACE = 4
} AUTHZ_SECURITY_ATTRIBUTE_OPERATION;

typedef enum _AUTHZ_SID_OPERATION {
    AUTHZ_SID_OPERATION_NONE = 0,
    AUTHZ_SID_OPERATION_REPLACE_ALL = 1,
    AUTHZ_SID_OPERATION_ADD = 2,
    AUTHZ_SID_OPERATION_DELETE = 3,
    AUTHZ_SID_OPERATION_REPLACE = 4
} AUTHZ_SID_OPERATION;

DWORD AuthzrFreeContext(
    [in, out] AUTHZR_HANDLE * ContextHandle);
DWORD AuthzrInitializeContextFromSid(
    [in] handle_t Binding,
    [in] DWORD Flags,
    [in] RPC_SID * Sid,
    [in] [unique] LARGE_INTEGER * pExpirationTime,
    [in] LUID Identifier,
    [out] AUTHZR_HANDLE * ContextHandle);
```

```
        DWORD AuthzrInitializeCompoundContext(
            [in] AUTHZR_HANDLE UserContextHandle,
            [in] AUTHZR_HANDLE DeviceContextHandle,
            [out] AUTHZR_HANDLE * CompoundContextHandle);
        DWORD AuthzrAccessCheck(
            [in] AUTHZR_HANDLE ContextHandle,
            [in] DWORD Flags,
            [in] AUTHZR_ACCESS_REQUEST * pRequest,
            [in] [range(1,16)] DWORD SecurityDescriptorCount,
            [in] [size_is(SecurityDescriptorCount)] SR_SD * pSecurityDescriptors,
            [in, out] AUTHZR_ACCESS_REPLY * pReply);
        DWORD AuthzGetInformationFromContext(
            [in] AUTHZR_HANDLE ContextHandle,
            [in] AUTHZ_CONTEXT_INFORMATION_CLASS InfoClass,
            [out] AUTHZR_CONTEXT_INFORMATION ** ppContextInformation);
        DWORD AuthzrModifyClaims(
            [in] AUTHZR_HANDLE ContextHandle,
            [in] AUTHZ_CONTEXT_INFORMATION_CLASS ClaimClass,
            [in] [range(1,65535)] DWORD OperationCount,
            [in] [size_is(OperationCount)] AUTHZ_SECURITY_ATTRIBUTE_OPERATION * pClaimOperations,
            [in] [unique] AUTHZR_SECURITY_ATTRIBUTES_INFORMATION * pClaims);
        DWORD AuthzrModifySids(
            [in] AUTHZR_HANDLE ContextHandle,
            [in] AUTHZ_CONTEXT_INFORMATION_CLASS SidClass,
            [in] [range(1,65535)] DWORD OperationCount,
            [in] [size_is(OperationCount)] AUTHZ_SID_OPERATION * pSidOperations,
            [in] [unique] AUTHZR_TOKEN_GROUPS * pSids);
    };
```

7 (Updated Section) Appendix B: Product Behavior

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

The terms "earlier" and "later", when used with a product version, refer to either all preceding versions or all subsequent versions, respectively. The term "through" refers to the inclusive range of versions. Applicable Microsoft products are listed chronologically in this section.

Windows Client releases

- Windows 8 operating system
- Windows 8.1 operating system
- Windows 10 operating system

Windows Server releases

- Windows Server 2012 operating system
- Windows Server 2012 R2 operating system
- Windows Server 2016 operating system
- Windows Server operating system
- Windows Server 2019 operating system

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

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

<1> Section 1.3: The RAZA client role is implemented on Windows 8 and later configured to run in a Windows domain environment. The RAZA server role is implemented on Windows Server 2012 and later and is operating when applicable Windows Server releases are configured to run in a Windows domain environment.

<2> Section 5.1: In Windows, this access is controlled by membership in the Windows Authorization Access Group.

8 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
7 Appendix B: Product Behavior	Added Windows Server 2019 to the list of applicable products.	Major

9 Index

A

C

D

E

S

T

V