[MS-WDSC]:

Windows Deployment Services Control 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
2/27/2009	0.1	Major	First Release.
4/10/2009	1.0	Major	Updated and revised the technical content.
5/22/2009	1.0.1	Editorial	Changed language and formatting in the technical content.
7/2/2009	1.0.2	Editorial	Changed language and formatting in the technical content.
8/14/2009	1.0.3	Editorial	Changed language and formatting in the technical content.
9/25/2009	1.1	Minor	Clarified the meaning of the technical content.
11/6/2009	1.1.1	Editorial	Changed language and formatting in the technical content.
12/18/2009	1.2	Minor	Clarified the meaning of the technical content.
1/29/2010	1.2.1	Editorial	Changed language and formatting in the technical content.
3/12/2010	1.2.2	Editorial	Changed language and formatting in the technical content.
4/23/2010	1.2.3	Editorial	Changed language and formatting in the technical content.
6/4/2010	1.3	Minor	Clarified the meaning of the technical content.
7/16/2010	1.3	None	No changes to the meaning, language, or formatting of the technical content.
8/27/2010	1.3	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2010	1.3	None	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	1.3	None	No changes to the meaning, language, or formatting of the technical content.
1/7/2011	1.3	None	No changes to the meaning, language, or formatting of the technical content.
2/11/2011	1.3	None	No changes to the meaning, language, or formatting of the technical content.
3/25/2011	1.3	None	No changes to the meaning, language, or formatting of the technical content.
5/6/2011	2.0	Major	Updated and revised the technical content.
6/17/2011	2.1	Minor	Clarified the meaning of the technical content.
9/23/2011	2.1	None	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	3.0	Major	Updated and revised the technical content.
3/30/2012	3.0	None	No changes to the meaning, language, or formatting of the technical content.
7/12/2012	3.0	None	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
10/25/2012	3.0	None	No changes to the meaning, language, or formatting of the technical content.
1/31/2013	3.0	None	No changes to the meaning, language, or formatting of the technical content.
8/8/2013	4.0	Major	Updated and revised the technical content.
11/14/2013	4.0	None	No changes to the meaning, language, or formatting of the technical content.
2/13/2014	4.0	None	No changes to the meaning, language, or formatting of the technical content.
5/15/2014	4.0	None	No changes to the meaning, language, or formatting of the technical content.
6/30/2015	5.0	Major	Significantly changed the technical content.
10/16/2015	5.0	None	No changes to the meaning, language, or formatting of the technical content.
7/14/2016	5.0	None	No changes to the meaning, language, or formatting of the technical content.
6/1/2017	5.0	None	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

The Windows Deployment Services (WDS) Control Protocol specifies an RPC interface that provides the ability to remotely invoke services provided by **WDS Server**. It is a client/server protocol which uses RPC as a transport. The protocol provides a generic invocation mechanism to send requests to a server and receive replies.

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 Glossary

This document uses the following terms:

authentication level: A numeric value indicating the level of authentication or message protection that **remote procedure call (RPC)** will apply to a specific message exchange. For more information, see [C706] section 13.1.2.1 and [MS-RPCE].

dynamic endpoint: A network-specific server address that is requested and assigned at run time. For more information, see [C706].

endpoint: A network-specific address of a remote procedure call (RPC) server process for remote procedure calls. The actual name and type of the endpoint depends on the **RPC** protocol sequence that is being used. For example, for RPC over TCP (RPC Protocol Sequence ncacn_ip_tcp), an endpoint might be TCP port 1025. For RPC over Server Message Block (RPC Protocol Sequence ncacn_np), an endpoint might be the name of a named pipe. For more information, see [C706].

Endpoint GUID: Set of relevant services provided by a **Service Provider** are grouped together and as a whole identified by a unique Endpoint GUID.

globally unique identifier (GUID): A term used interchangeably with **universally unique identifier (UUID)** in Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the value. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the **GUID**. See also **universally unique identifier (UUID)**.

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

OpCode: Each service provided by Service Provider under an **Endpoint GUID** is identified by a number which must be unique under that **Endpoint GUID**.

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

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

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

security provider: A Component Object Model (COM) object that provides methods that return custom information about the security of a site.

service provider: A module that is loaded by the WDS Server and is responsible for providing services to the clients.

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.

WDS server: A Windows Deployment Services (WDS) server that communicates with clients by using the WDS OS Deployment Protocol to aid in deployment of an OS image on a client machine. Clients also communicate to a WDS server to request initiation/setup of multicast sessions for content available in multicast namespace on server.A WDS server provides an extensible mechanism to allow **service providers** to provide services to clients.

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

[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

[RFC4122] Leach, P., Mealling, M., and Salz, R., "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, July 2005, http://www.rfc-editor.org/rfc/rfc4122.txt

1.2.2 Informative References

None.

1.3 Overview

Windows Deployment Services (WDS) Control Protocol is a generic client/server protocol which is used to invoke services provided by **Service Providers** in WDS Server. The WDS Control Protocol is a simple protocol with no state shared across multiple calls. Each call is considered one complete request.

A typical service invocation involves the following:

1. The client has already obtained the name of the **WDS Server**, **Endpoint GUID** for Service Provider and **OpCode** for the operation being invoked.
2. The client constructs a request by packaging required variables (as specified in section 2.2.1), Endpoint GUID and OpCode.
3. The WDS Control Protocol sends the request to the server by using **RPC** interface (as specified in section 3).
4. The WDS Server dispatches the request to the appropriate Service Provider based on Endpoint GUID.
5. Based on the Endpoint GUID and OpCode in the request, Service Provider will:
   - Validate that the client has appropriate rights to perform the operation.
   - Unpack the variables stored in the packet.
   - Perform the requested operation.
   - Package the results in pre-determined variables and complete the RPC request.
6. The client will check for success or failure of the request (as specified in section 3.1.4.2 and 3.2).
7. Unpack the variables from the reply packet and process the results.

The following diagram shows a client making a request to the WDS Server:

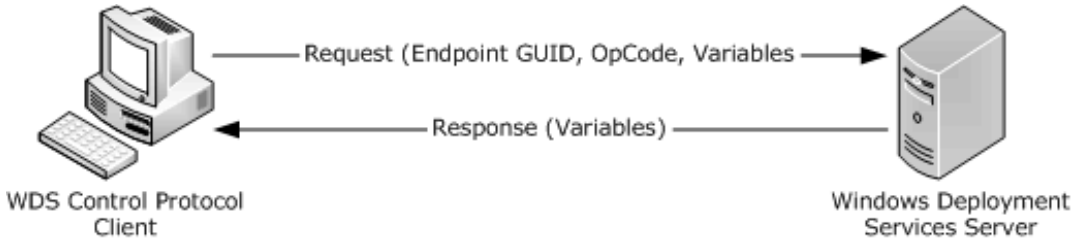


Figure 1: Client to server request

1.4 Relationship to Other Protocols

The WDS Control Protocol relies on **RPC**, as defined in [MS-RPCE], as the transport. It uses RPC to send the request to the **WDS Server** and receive replies.

The following diagram illustrates the relationship of WDS Control Protocol and how it relates to RPC:

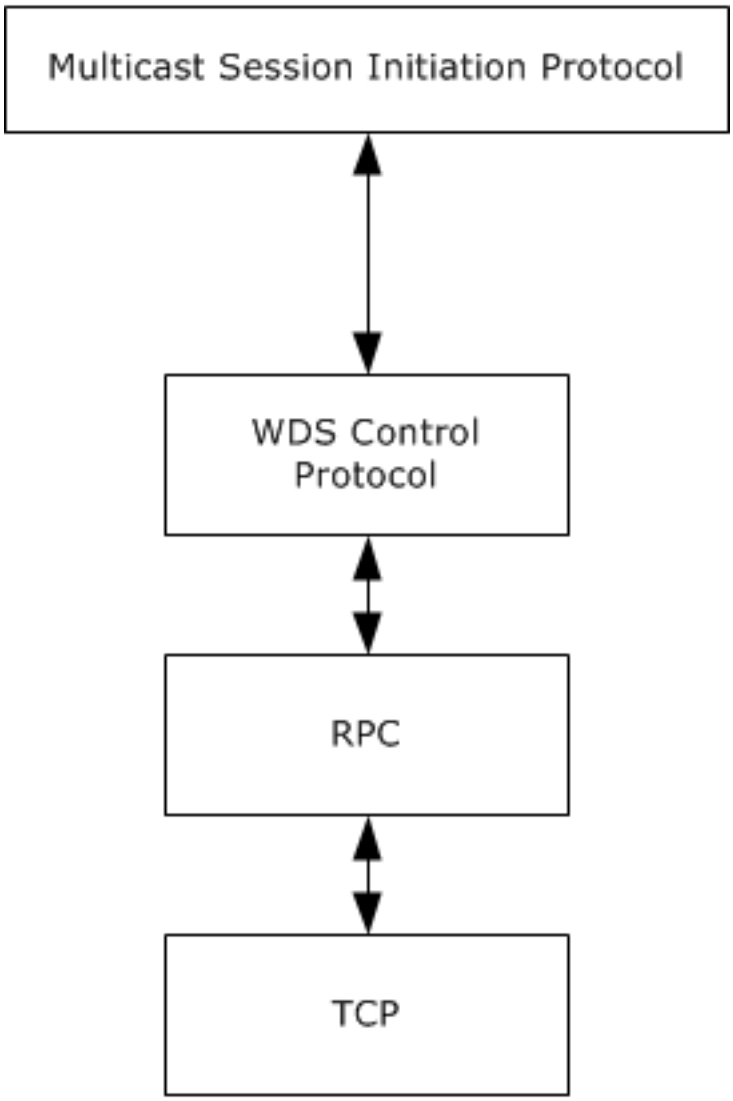


Figure 2: Relationship to other protocols

1.5 Prerequisites/Preconditions

This protocol is implemented on top of **RPC** and, as a result, has the prerequisites identified in [MS-RPCE].

The WDS Control Protocol assumes that a client has obtained the name of the server that supports this protocol, as well as an **Endpoint GUID**, security (authenticated or unauthenticated) requirements, an **OpCode**, variables required for each service the client intends to invoke, and variables used by these services to return the results.

1.6 Applicability Statement

This protocol is applicable when an application needs to invoke services provided by Service Providers residing on **WDS Server**.

1.7 Versioning and Capability Negotiation

This document covers versioning issues in the following areas:

- Supported Transports: This protocol uses single **RPC Protocol Sequence**.
- Protocol Versions: This protocol uses a single **RPC** interface.
- Security and Authentication Methods: Authentication and security are provided as specified in [MS-RPCE].
- Localization: This protocol acts as pass-thru for all strings with no support for localization built into the protocol.
- Capability Negotiation: The WDS Control Protocol does not support negotiation of the interface version to use. Instead, this protocol uses only the interface version number specified in the **IDL** for versioning and capability negotiation.

1.8 Vendor-Extensible Fields

Vendors can add new Service Providers by generating a new **globally unique identifier (GUID)** as specified in [RFC4122], and exposing new **OpCodes** for the Service Providers.

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

Parameter	Value	Reference
RPC Interface UUID for WDS Control Protocol	1A927394-352E-4553-AE3F-7CF4AAFCA620	[C706] section A.2.5

2 Messages

2.1 Transport

All implementations MUST support **RPC** over TCP with **dynamic endpoints**.

The protocol MUST use the following **UUID**: 1A927394-352E-4553-AE3F-7CF4AAFCA620.

WDS Control Protocol uses RPC to transport the request packet from client to **WDS Server** and to transport the reply packet back to the client.

2.1.1 Server Security Settings

The WDS Control Protocol uses Security Service Provider (SSP) security provided by **RPC** as specified in [MS-RPCE] for sessions. The server SHOULD register the following SSPs, as supported **security providers**:

- RPC_C_AUTHN_GSS_NEGOTIATE
- RPC_C_AUTHN_GSS_KERBEROS
- RPC_C_AUTHN_WINNT

The **WDS Server** MUST allow authenticated and unauthenticated access to RPC clients. The WDS Server SHOULD support the maximum number of concurrent calls supported by RPC, as specified in [MS-RPCE] section 3.3.1.5.8.

2.1.2 Client Security Settings

The client MUST choose the security settings as required by the **Service Provider** for a given **Endpoint GUID**. An Endpoint GUID is a unique **GUID** used to identify a set of **RPC** services provided by a Service Provider that are grouped together as a relevant whole. All services under an Endpoint GUID for a Service Provider share the same security requirements. A Service Provider MAY expose multiple Endpoint GUIDs and each MAY have different security requirements.

The WDS Control Protocol RPC client MUST use SSP security provided by RPC as specified in [MS-RPCE] when invoking a service of a Service Provider that requires authenticated clients for the Endpoint GUID. A client SHOULD authenticate using RPC_C_AUTHN_GSS_NEGOTIATE.

A client communicating to **WDS Server** using authentication MUST use RPC_C_AUTHN_LEVEL_PKT_PRIVACY. An unauthenticated client SHOULD use RPC_AUTHN_LEVEL_NONE.

2.1.3 RPC as Transport

WDS Control Protocol uses **RPC** to transport a packet to the **WDS Server**. The request packet acts as the input parameter for the RPC function call. The reply packet from WDS Server is transported back to the client as an output parameter of the RPC function call.

2.2 Common Data Types

The base types used by the WDS Control Protocol are defined as the **RPC** base types. Additional data types are specified in [C706] and [MS-RPCE].

2.2.1 Messages

The following diagram illustrates the overall structure of request and reply packets.

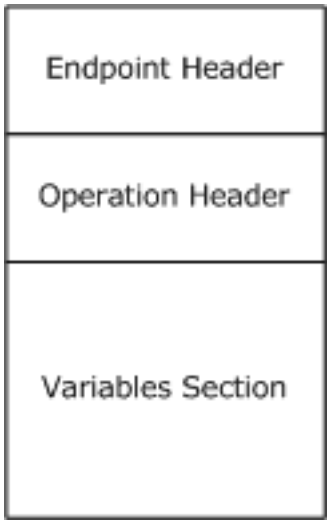


Figure 3: Overall packet structure

The Endpoint Header is defined in section 2.2.1.1 and specifies the target **Endpoint GUID**.

The Operation Header is defined in section 2.2.1.2. For request packets, this header specifies the target operation (**OpCode**) to perform. For reply packets, this header specifies the results of the operation.

The Variables Section is defined in section 2.2.1.3. For request packets, this section MUST specify the variables required by the service being invoked on the **WDS Server**. For reply packets, this section MUST specify the variables as expected by the client for the service invoked on the WDS Server.

All multibyte values specified in this document are in little-endian format unless specified otherwise.

2.2.1.1 Endpoint Header

The **Endpoint** Header is defined as follows:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Size-Of-Header																Version															
Packet-Size																															
Endpoint-GUID (16 bytes)																															
...																															
...																															
Reserved (16 bytes)																															

...
...

Size-Of-Header (2 bytes): MUST be set to the size of header in bytes, which is 0x0028.

Version (2 bytes): MUST be set to 0x0100.

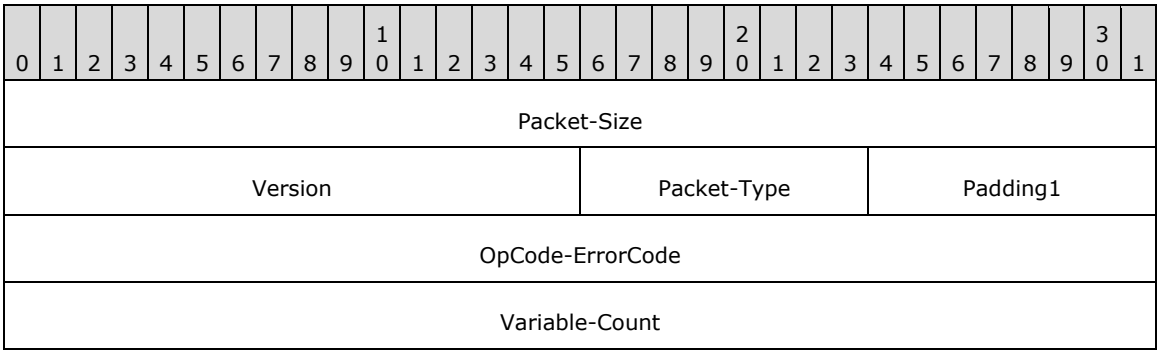
Packet-Size (4 bytes): MUST be set to the number of bytes being transmitted for the packet. It MUST be the arithmetic sum of the sizes in bytes of Endpoint Header, Operation Header, and Variables Section.

Endpoint-GUID (16 bytes): MUST be set to the **Endpoint GUID**, as specified in [MS-DTYP] section 2.3.4.2, of the Service Provider, which provides the service to be invoked.

Reserved (16 bytes): MUST be set to zeros.

2.2.1.2 Operation Header

The Operation Header is defined as follows:



Packet-Size (4 bytes): MUST be set to the arithmetic addition of the sizes in bytes of the Operation Header and Variables Section.

Version (2 bytes): MUST be set to 0x0100.

Packet-Type (1 byte): SHOULD be set to value as defined in section 2.2.1.2.1.

Padding1 (1 byte): Padding MAY be set to zero and MUST be ignored by receiver.

OpCode-ErrorCode (4 bytes): For request packets, this field MUST be set to the **OpCode** for the service being invoked on the **WDS Server**. The **Endpoint GUID** combined with the OpCode uniquely identifies the service to be invoked on the WDS Server. For reply packets, this field specifies the error code for the invoked service. Section 3.1.4.2 specifies in detail how operation results are communicated to the client.

Variable-Count (4 bytes): MUST set to the number of variables in the Variables Section.

2.2.1.2.1 Packet Type

The Packet Type field in the Operation Header SHOULD be set to the value in the following table. The receiver MAY validate the Packet Type before accepting the packet.<1>

Packet Type	Description
WDSCPL_PACKET_REQUEST 0x01	The client SHOULD set this value for all request packets. **WDS Server** MAY validate the packet type before accepting the packet.
WDSCPL_PACKET_REPLY 0x02	The reply packet from WDS Server SHOULD set the packet type to this value. The client SHOULD NOT reject a packet if the packet type is not properly set.

2.2.1.3 Variables Section

The Variables Section in the request packet is used to specify parameters required by the service being invoked on **WDS Server**. In reply packets from WDS Server, the variables are used to communicate and provide output of the operation performed by Service Provider on WDS Server. The count of variables in the Variables Section MUST match the count of variables specified in the Operation Header. Each variable is defined using Variable Description Block as defined in section 2.2.1.3.1.

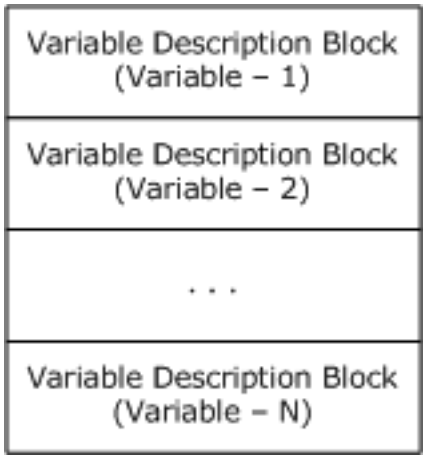
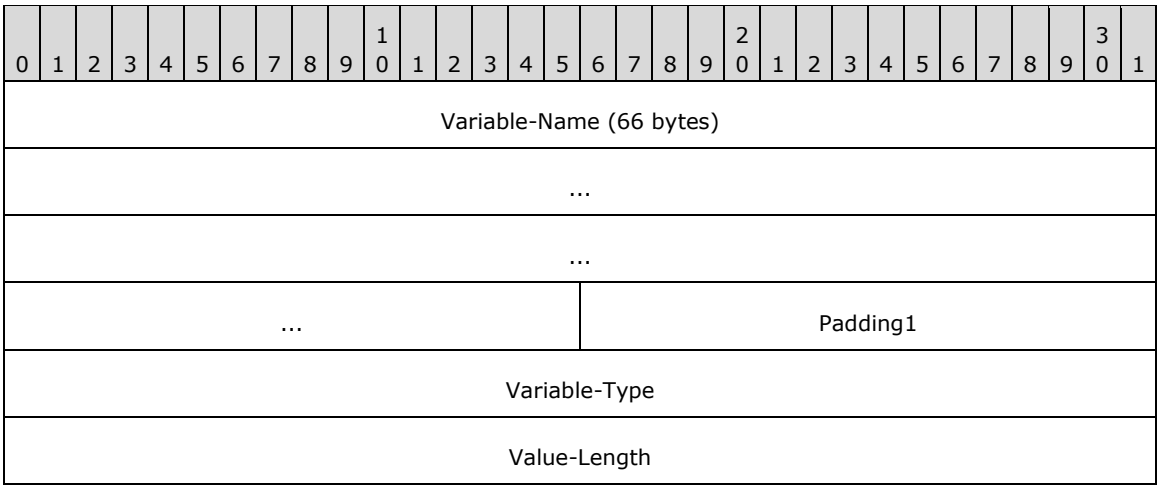


Figure 4: Variables section

2.2.1.3.1 Variable Description Block

The Variable Description Block is defined as follows:



Array-Size
Variable-Value (variable)
...
Padding_end (variable)
...

Variable-Name (66 bytes): Specifies the name of the variable as a Unicode string. The variable name MUST end with a null character. Duplicate variable names MUST NOT be allowed. **Variable-Name** MUST be treated as case-insensitive.

Padding1 (2 bytes): Padding MAY be set to zero and MUST be ignored by receiver.

Variable-Type (4 bytes): MUST be set to the data type of the variable as specified in section 2.2.1.3.2.

Value-Length (4 bytes): This field MUST be set as specified in section 2.2.1.3.3.

Array-Size (4 bytes): This field MUST be set as specified in section 2.2.1.3.3.

Variable-Value (variable): Specifies the value for the variable.

Padding_end (variable): Padding MAY be set to zero and MUST be ignored by receiver. Padding_end size MUST be such that the total size (in bytes) of the Variable Description Block is evenly divisible by 16.

2.2.1.3.2 Variable Types

WDS Control Protocol supports multiple data types. The identifier for each data type is a four bytes unsigned numeric value. The identifier for data type consists of the base data type and set of optional type modifiers.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Base Type																Type Modifier															

If a data type does not have a type modifier then the type modifier MUST be set to zero.

2.2.1.3.2.1 Base Types

WDS Control Protocol supports following base data types:

Base Type	Size of Value in Bytes	Fixed Length?	Description
WDSCPL_VAR_BYTE 0x0001	1	Yes	Single byte unsigned value
WDSCPL_VAR_USHORT 0x0002	2	Yes	Specifies two byte unsigned numeric value
WDSCPL_VAR_ULONG	4	Yes	Specifies four byte unsigned numeric value

Base Type	Size of Value in Bytes	Fixed Length?	Description
0x0004			
WDSCPL_VAR_ULONG64 0x0008	8	Yes	Specifies an eight byte unsigned numeric value
WDSCPL_VAR_STRING 0x0010	Variable length	No	Specifies a CHAR ([MS-DTYP] section 2.2.7) string which MUST be terminated by a zero (null) character.
WDSCPL_VAR_WSTRING 0x0020	Variable length	No	Specifies a UNICODE ([MS-DTYP] section 2.2.56) string which MUST be terminated by a zero (null) character.
WDSCPL_VAR_BLOB 0x0040	Variable length	No	Specifies variable length stream of bytes.

2.2.1.3.2.2 Type Modifiers

Each data type (as specified in section 2.2.1.3.2.1) can have a type modifier associated with it. The following list specifies the list of available type modifiers:

Type Modifier	Description
WDSCPL_VAR_ARRAY 0x1000	Specifies that the value contains an array of base data type.

2.2.1.3.3 Variable Value Length

The count of bytes that hold the value for a variable in Variable Description Block are computed using the **Value-Length** and **Array-Size** fields of the Variable Description Block.

2.2.1.3.3.1 Variables without Type Modifiers

For data types without any type modifiers, the **Array-Size** field SHOULD be set to zero.

For fixed length data type the **Value-Length** field SHOULD be set to the size in bytes as specified in section 2.2.1.3.2.1.

For variable length data types the **Value-Length** field MUST specify the total number of bytes containing the value.

2.2.1.3.3.2 Variables with WDSCPL_VAR_ARRAY Type Modifier

For fixed length data types the **Value-Length** field SHOULD be set to the size in bytes as specified in section 2.2.1.3.2.1.

For variable length data types the **Value-Length** field MUST specify the total number of bytes containing the value.

The **Array-Size** field specifies the number of elements contained in the array and MUST NOT be zero.

The total number of bytes that hold the value for a variable are obtained by arithmetic multiplication of **Value-Length** and **Array-Size**.

The individual elements of the array are accessed using a 0-based index with a maximum value for the index being (**Array-Size** - 1).

To access a value at a specified index, multiply the **Value-Length** by the index to obtain an offset into the **Variable-Value** field. The **Value-Length** field specifies the number of bytes to read at the derived offset in order to read the value at that index.

3 Protocol Details

3.1 Server Details

This section specifies the WDS Server Control Protocol behavior.

3.1.1 Abstract Data Model

This section describes a conceptual model for handling the WDS Control Protocol by a WDS Server. The described model is provided to explain how the protocol behaves. Implementations are not required to adhere to this model as long as their external behavior is consistent with that described in this document.

Global Server State: The global state of the server, set to one of the following values:

- Loading: the **WDS Server** is loading and initializing the Service Providers.
- Running: WDS Server has loaded and initialized all Service Providers and serving client requests.
- Stopping: WDS Server is shutting down.
- Service Provider: A **Service Provider** is a module that is loaded by WDS Server based on configuration. A Service Provider registers the **Endpoint GUID** with the WDS Server so that incoming requests can be routed.
- Configuration: A list of Service Providers that are to be loaded by WDS Server during initialization.
- Endpoint GUID: An Endpoint GUID is registered by a Service Provider with the WDS Server. The details provided by a Service Provider when registering an Endpoint GUID can be found in section 3.1.3.1.
- OpCode: An **OpCode** uniquely identifies a service offered by a Service Provider under a registered Endpoint GUID.

3.1.1.1 Configuration

The list of names used as (name, value) pair in **WDS Server** Configuration information are given below:

ServiceProviderList: A list of modules that will be loaded during WDS Server initialization. Each module represents a Service Provider.

3.1.2 Timers

No timers are required for **WDS Server** beyond those used internally by **RPC** to implement resiliency to network outages, as specified in [MS-RPCE] section 3.2.3.2.1.

3.1.3 Initialization

At initialization time, **WDS Server** MUST initialize all registered Service Providers and **RPC** server.

3.1.3.1 Service Providers Initialization

WDS Server: MUST read the configuration (as defined in section 3.1.1.1) and initialize each **Service Provider**. Each Service Provider MUST register all **Endpoint GUIDs** it is to offer clients using WDS Server. Each registered Endpoint GUID MUST include the following configuration information:

GUID: A 128 bit value, which uniquely identifies that Endpoint GUID.

SecurityAccess: MUST specify if WDS Server allows authenticated and/or unauthenticated clients to invoke services offered by Service Provider under this Endpoint GUID.

3.1.3.2 RPC Server Initialization

The **WDS Server** Control Provider server MUST register the **RPC** interface and begin listening on the **RPC transport** as specified in section 2.1.

3.1.4 Message Processing Events and Sequencing Rules

Methods in RPC Opnum Order

Method	Description
WdsRpcMessage	This function is used to transport the request packet to the server and to bring the reply packet. Opnum: 0

3.1.4.1 WdsRpcMessage (opnum 0)

The WdsRpcMessage (opnum 0) method sends the request packet to the server and returns the corresponding reply packet.

```
unsigned long WdsRpcMessage(
  handle_t hBinding,
  [in] unsigned long uRequestPacketSize,
  [in, size_is(uRequestPacketSize)]
    byte bRequestPacket[],
  [out] unsigned long* puReplyPacketSize,
  [out, size_is(, *puReplyPacketSize)]
    byte** pbReplyPacket
);
```

hBinding: The RPC binding handle that the client established when it connected to the server.

uRequestPacketSize: The client MUST pass the total size of request packet in bytes.

bRequestPacket: A pointer to the request packet. The packet MUST be constructed as specified in section 2.2.1.

puReplyPacketSize: The **WDS Server** MUST set this to the total size of the reply packet in bytes.

pbReplyPacket: The WDS Server MUST set this to the reply packet. The reply packet MUST be constructed as specified in section 2.2.1.

Return Values: The method MUST return ERROR_SUCCESS (0x00000000) on success or a non-zero Win32 error code value if an error occurred.

When processing this call, the WDS Server MUST do the following:

- If the Global Server State is not Running (section 3.1), the server MUST return a failure.
- Server MUST validate the Endpoint Header (section 2.2.1.1) and extract the **Endpoint GUID**, and if invalid, the server MUST return a failure.

- Server MUST search through the list of registered Endpoint GUIDs to match the Endpoint GUID specified in the Endpoint Header. If no match is found, the server MUST return a failure.
- Server MUST query the **authentication level** of the **RPC** call and validate as follows:
  - If registered Endpoint GUID allows both authenticated and unauthenticated clients, then continue to next step.
  - If the registered Endpoint GUID requires an authenticated client and client RPC call is unauthenticated, the server MUST return a failure.
  - If the registered Endpoint GUID requires an unauthenticated client and the client RPC call is authenticated, the server MUST return a failure.
  - For authenticated clients, server MUST enforce RPC_C_AUTHN_LEVEL_PKT_PRIVACY and return failure if client is not using it.
- Server MUST dispatch the request packet to the Service Provider that registered the Endpoint GUID.
- Service Provider MUST validate the Operation Header (section 2.2.1.2) and extract the **OpCode**. The server MUST return failure if invalid.
- Service Provider MUST validate that it offers the requested OpCode under the registered Endpoint, and if invalid the server MUST return failure.
- Service Provider MUST validate and extract all variables from the Variables Section (section 2.2.1.3), and return failure if invalid.
- For the requested OpCode, Service Provider MUST validate that all required variables are present.
- Service Provider MAY validate that the client is authorized to perform the requested operation.
- Service Provider MUST perform the requested operation as identified by the Endpoint GUID and OpCode from the request packet. If an error is encountered, it MUST be handled as specified in section 3.1.4.2.
- Service Provider MUST construct the reply packet (section 2.2.1) and hand it to WDS Server.
- WDS Server MUST complete the RPC call setting the output parameters for the function to the packet provided by Service Provider.

3.1.4.2 Failure Cases

WDS Server MUST return a failure status for the **RPC** call if it encounters any failure before passing the request to a Service Provider, or after the reply packet has been handed to WDS Server by a Service Provider.

Service Providers SHOULD report failures by instructing the WDS Server to return the appropriate failure code for the RPC call.

Service Providers MAY construct a reply packet and set the failure error code in the Operation Header (section 2.2.1.2). In this case, the RPC call succeeds, but the OpCode-ErrorCode field in the Operation Header MUST contain the appropriate error code.<2>

3.1.5 Timer Events

None.

3.1.6 Other Local Events

None.

3.2 Client Details

3.2.1 Abstract Data Model

This section describes a conceptual model of possible handling of WDS Control Protocol by clients to participate in this protocol. The described model 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.

3.2.2 Timers

The clients MAY request a timeout for the operation when using WDS Control Protocol. If a timeout is specified and a request to **WDS Server** does not complete before the timeout, then WDS Control Protocol MUST cancel the **RPC** call and return appropriate failure.

If client does not request a timeout, then WDS Control Protocol requires no timers beyond those used internally by RPC to implement resiliency to network outages, as specified in [MS-RPCE] section 3.2.3.2.1.

3.2.3 Initialization

None.

3.2.4 Message Processing Events and Sequencing Rules

The client MUST provide information to WDS Control Protocol to uniquely identify the service to be invoked. It MUST include all variables required by service being invoked. The following is the list of information to be provided:

- Server name or IP address
- **Endpoint GUID**
- Security (authenticated or unauthenticated call)
- **OpCode**
- Required variables

The client MAY also specify a timeout for the **RPC** call.

When processing the request, the client MUST follow these steps:

1. Construct the request packet as specified in section 2.2.1.
2. Set the **authentication level** for the RPC call (either authenticated or unauthenticated).
3. Set the timeout for RPC call if one is specified by client.
4. Send request to server as specified in section 3.1.4.1.

If call executes and returns with success code of ERROR_SUCCESS (0x00000000), the WDS Control Protocol MUST do the following:

1. Validate the Endpoint Header (section 2.2.1.1), Operation Header (section 2.2.1.2) and Variables Section (section 2.2.1.3).

2. The Endpoint GUID in the Endpoint Header MUST match the Endpoint GUID specified by client.

3. If the **OpCode-ErrorCode** field in the Operation Header is not set to error code ERROR_SUCCESS (0x00000000), the WDS Control Protocol MAY return the same error code to the client.

3.2.5 Timer Events

The timer event is start just before the **RPC** call is initiated. If the timer expires and the RPC call has still not completed, the WDS Control Protocol MUST cancel the RPC call and return the appropriate error code to the client.

3.2.6 Other Local Events

None.

4 Protocol Examples

The following diagram illustrates a client invoking a service provided by Service Provider on a WDS Server:

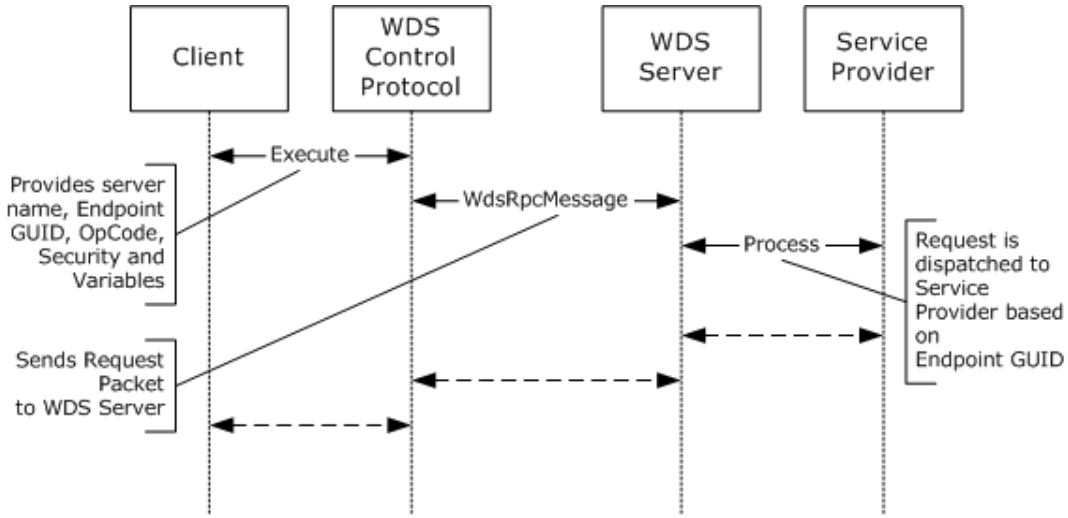


Figure 5: Client invoking a service

5 Security

5.1 Security Considerations for Implementers

None.

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.

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

[
    uuid(1A927394-352E-4553-AE3F-7CF4AAFCA620),
    version(1.0)
]
interface WdsRpcInterface
{
    unsigned long
    WdsRpcMessage(
      handle_t hBinding,
      [in] unsigned long uRequestPacketSize,
      [in, size_is(uRequestPacketSize)] byte bRequestPacket[],
      [out] unsigned long* puReplyPacketSize,
      [out, size_is(, *puReplyPacketSize)] byte** pbReplyPacket

    );
}
```

7 Appendix B: Product Behavior

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

- Windows Server 2003 operating system
- Windows Server 2008 operating system
- Windows Server 2008 R2 operating system
- Windows Server 2012 operating system
- Windows Server 2012 R2
- Windows Server 2016 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

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

<1> Section 2.2.1.2.1: The PacketType field is not properly set in the reply packets.

<2> Section 3.1.4.2: When an operation fails, some operations can set the OpCode-ErrorCode field to the failure error code.

8 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

9 Index

S

T

V

W