[MC-DPLHP]:
DirectPlay 8 Protocol:
Host and Port Enumeration Specification

This protocol documentation is intended for use in conjunction with publicly available standard specifications, network programming art, and Microsoft Windows distributed systems concepts, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

A protocol specification 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.

Revision Summary

Date	Revision History	Revision Class	Comments
08/10/2007	0.1	Major	Initial Availability
09/28/2007	0.2	Minor	Updated the technical content.
10/23/2007	0.2.1	Editorial	Revised and edited the technical content.
11/30/2007	1.0	Major	Updated and revised the technical content.

Date	Revision History	Revision Class	Comments
01/25/2008	2.0	Major	Updated and revised the technical content.

Table of Contents

1 Introduction

This document specifies the DirectPlay 8 Protocol: Host and Port Enumeration.

The DirectPlay 8 Protocol: Host and Port Enumeration allows a **DirectPlay 8 client application** to discover one or more **DirectPlay 8 server applications**.

1.1 Glossary

DirectPlay 8 Application: A software process that communicates with one or more software processes over a communications network by using the DirectPlay 8 family of protocols.

DirectPlay 8 Client Application: A **DirectPlay 8 application** seeking to connect to another **DirectPlay 8 application** that is hosting a **DirectPlay 8 session**. When connected, the actual communication between nodes in a **DirectPlay 8 session** may be client/server or peer to peer. The term "client" in this definition is meant to indicate the role that the **DirectPlay 8 client application** is taking in the host enumeration process, which is the **DirectPlay 8 application** that is seeking to find and connect to a host of a **DirectPlay 8 session**.

DirectPlay 8 Protocol: This protocol should be implemented on top of UDP, as specified in [MC-DPL8R] section 1.7, but may be implemented on top of any connectionless, datagram-oriented protocol. The **DirectPlay 8 Protocol** allows for the delivery of reliable and non-reliable, as well as sequenced and non-sequenced messages. Each message is constructed such that at least one bit is set in the first byte. When a message is received and the first byte is zero, the entire message is passed through for processing as described in this specification [MC-DPLHP]. Otherwise, the message is processed as described in [MC-DPL8R].

DirectPlay 8 Server Application: A **DirectPlay 8 application** that is hosting a **DirectPlay 8 session**. When connected, the actual communication between nodes in a **DirectPlay 8 session** may be client/server or peer to peer. The term "server" in this definition is meant to indicate the role that the **DirectPlay 8 server application** is taking in the host enumeration process, which is the **DirectPlay 8 application** that is currently hosting a **DirectPlay 8 session**.

DirectPlay 8 Service Provider: A service provider that may be implemented on top of the **DirectPlay 8 Protocol**[MC-DPL8R], as specified in the DirectPlay 8 Protocol: Core and Service Providers Specification [MC-DPL8CS]. When a message is passed through for processing, the protocol [MC-DPLHP] interacts directly with the **DirectPlay 8 Service Provider**.

DirectPlay 8 Session: A stateful communications session that uses the **DirectPlay 8 session** management protocol, as specified in [MS-DPDX].

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

1.2 References

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. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[IANAPORT] Internet Assigned Numbers Authority, "Port Numbers", November 2006, http://www.iana.org/assignments/port-numbers

[MC-DPL8CS] Microsoft Corporation, "DirectPlay 8 Protocol: Core and Service Providers Specification" September 2007.

[MC-DPL8R] Microsoft Corporation, "DirectPlay 8 Protocol: Reliable Specification" September 2007.

[MS-DPDX] Microsoft Corporation, "DirectPlay DXDiag Usage Protocol Specification", September 2007.

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980, http://www.ietf.org/rfc/rfc768.txt

1.2.2 Informative References

[MS-DTYP] Microsoft Corporation, "Windows Data Types", January 2007.

1.3 Protocol Overview (Synopsis)

The DirectPlay 8 Protocol: Host and Port Enumeration enables a DirectPlay 8 client application to discover DirectPlay 8 server applications.

The basic functionality is simple. A DirectPlay 8 client application sends an EnumQuery message over a communications network. If the EnumQuery message is received by a DirectPlay 8 server application, and the DirectPlay 8 server application wishes to enable the DirectPlay 8 client application to connect to the **DirectPlay 8 session** that it is hosting, it replies to the DirectPlay 8 client application with an EnumResponse message. The EnumResponse message contains the information required by the DirectPlay8 client application to attempt to connect to the DirectPlay 8 session being hosted by the DirectPlay 8 server application.

Note that it is possible for one EnumQuery message to be delivered to multiple DirectPlay 8 server applications, each of which may or may not choose to reply with an EnumResponse message. Therefore, one EnumQuery message may generate zero, one, or more than one EnumResponse messages. The DirectPlay 8 client application is not obligated to connect to any of the DirectPlay 8 server applications that reply with an EnumResponse message. On the contrary, one of the purposes of the DirectPlay 8 Protocol: Host and Port Enumeration process is to allow a DirectPlay 8 client application to discover multiple DirectPlay 8 sessions, and choose which one to join based on application-specific preferences such as game modes, latency, number of players, user preferences, and so on.

EnumQuery and EnumResponse messages are delivered using the **DirectPlay 8 Service Providers** directly. These messages do not use the **DirectPlay 8 Protocol**. DirectPlay 8 Service Providers do not provide reliable message delivery. This means that both EnumQuery and EnumResponse messages may be lost. It is therefore expected that a DirectPlay 8 client application will send multiple EnumQuery requests while searching for available DirectPlay 8 server applications.

It is possible, although not required, for the DirectPlay 8 client application to note the round trip latency of each EnumQuery/EnumResponse pair, and the number of EnumQuery/EnumResponse pairs that are lost, and use that information to predict the future quality of the network service between itself and the responding DirectPlay 8 server applications.

1.4 Relationship to Other Protocols

How a DirectPlay 8 client application connects to the DirectPlay 8 session being hosted by a DirectPlay 8 server application that chooses to send an EnumResponse message is specified in [MS-DPDX].

EnumQuery and EnumResponse messages are directly delivered via UDP using DirectPlay 8 pass-through functionality [MC-DPL8R] and are not provided with reliable message delivery. As a result, both EnumQuery and EnumResponse messages may be lost.

A DirectPlay 8 Service Provider allows DirectPlay 8 messages to be layered on top of multiple different underlying network transport protocols, such as IPv4, IPv6, IPX, and Serial links. The details of the DirectPlay 8 Service Provider are specified in [MC-DPL8CS].

1.5 Prerequisites/Preconditions

The DirectPlay 8 client application and DirectPlay 8 server application must have already agreed upon the Application GUID they will use to identify themselves as instances of the same **DirectPlay 8 application**.

The DirectPlay 8 server application must be hosting a DirectPlay 8 session before it can participate in the DirectPlay 8 Protocol: Host and Port Enumeration.

1.6 Applicability Statement

The DirectPlay 8 Protocol: Host and Port Enumeration is appropriate for use when a DirectPlay 8 client application needs to query multiple DirectPlay 8 server applications for their current status, to determine (possibly with the assistance of user input) which DirectPlay 8 server application to connect to, if any.

On IPv4 networks it is also appropriate to use the DirectPlay 8 Protocol: Host and Port Enumeration when only the IPv4 address information of a DirectPlay 8 server application is known, and the DirectPlay 8 client application needs to discover which port the DirectPlay 8 server application is using. In this case, the DirectPlay 8 client application should send the EnumQuery (section 2.2.1) message to the DirectPlay 8 server well-known port, as specified in [IANAPORT]. Note that not all DirectPlay 8 server applications will choose to respond to EnumQuery messages sent to this port, nor will all implementations of this protocol support the use of the DirectPlay 8 server well-known port.

1.7 Versioning and Capability Negotiation

The DirectPlay 8 Protocol: Host and Port Enumeration has no versioning or capability negotiation features. However, the application may use the application-specific fields of the protocol to perform application-level versioning or capability negotiation.

1.8 Vendor-Extensible Fields

There are no vendor extensible fields in the DirectPlay 8 Protocol: Host and Port Enumeration.

1.9 Standards Assignments

DirectPlay 8 uses the following well-known TCP and UDP port assignments, as specified in [IANAPORT].

Parameter	Value	Used By
directplay8	6073/tcp	DirectPlay 8
directplay8	6073/udp	DirectPlay 8
directplaysrvr	47624/tcp	DirectPlay Server
directplaysrvr	47624/udp	DirectPlay Server

In addition to port 6073 and 47624, a DirectPlay 8 application may also use any other arbitrary port for "in-game" communication. However, DirectPlay 8 does not mandate that these other ports be numbered within any particular range, or selected according to any particular scheme. In fact, the DirectPlay 8 Host and Port Enumeration Protocol primarily uses port 6073 to allow for discovery of these other ports.

The sender of a query message may use any port for the source of the message. The server listening on port 6073 will reply to the address and port from which it receives a query. While a DirectPlay 8 application may find it convenient to send a query from the port that is being used for "in-game" communication, the sender is not required to use this port or any other particular port.

Although many DirectPlay 8 applications explicitly specify the port numbers to use for "in-game" communication, when the application has not specified particular port number(s), the DirectPlay 8 implementation chooses the first available port in the range 2302 - 2400.

2 Messages

The following sections specify how DirectPlay 8 Protocol: Host and Port Enumeration messages are transported and DirectPlay 8 Protocol: Host and Port Enumeration message syntax.

2.1 Transport

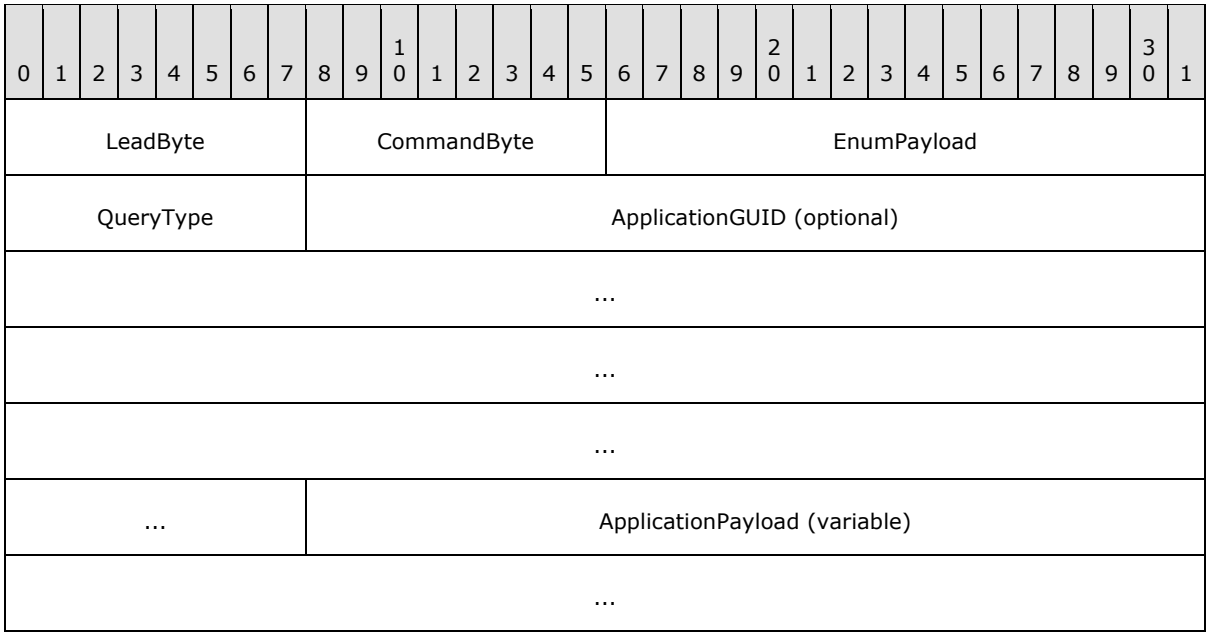
EnumQuery and EnumResponse messages are delivered using the DirectPlay 8 Service Providers directly. These messages do not use the DirectPlay 8 Protocol. DirectPlay 8 Service Providers do not provide reliable message delivery. This means that both EnumQuery and EnumResponse messages may be lost.

2.2 Message Syntax

2.2.1 EnumQuery

The EnumQuery message is used to query for instances of a DirectPlay 8 server application that is hosting a DirectPlay 8 session.

The size of the variable length field in the EnumQuery message is limited by whatever limit is placed on the overall message size by the DirectPlay 8 Service Provider that is used to transmit the message.



LeadByte (1 byte): This field is 8 bits in length. It MUST be 0x00.

CommandByte (1 byte): This field is 8 bits in length. It MUST be 0x02.

EnumPayload (2 bytes): This field is 16 bits in length. The **EnumPayload** is a value selected by the sender of the EnumQuery message that can be used to match EnumResponse messages to their corresponding EnumQuery.

QueryType (1 byte): This field is 8 bits in length.

Value	Meaning
0x00	Indicates that this EnumQuery message contains no **ApplicationGUID** field. All DirectPlay 8 server applications that receive this EnumQuery MAY respond to it.
0x01	Indicates that this query contains an **ApplicationGUID** field. Only DirectPlay 8 server applications that are identified by the **ApplicationGUID** MAY respond to this EnumQuery.

ApplicationGUID (16 bytes): This optional field, when present, is 128 bits in length.

Value	Meaning
0x01	The field contains the GUID of the DirectPlay 8 server application type that is being queried. Only DirectPlay 8 server applications that match this GUID MAY respond.
0x02	The field is not present in the message. For more information about the GUID type, see [MS-DTYP] section 2.3.2.

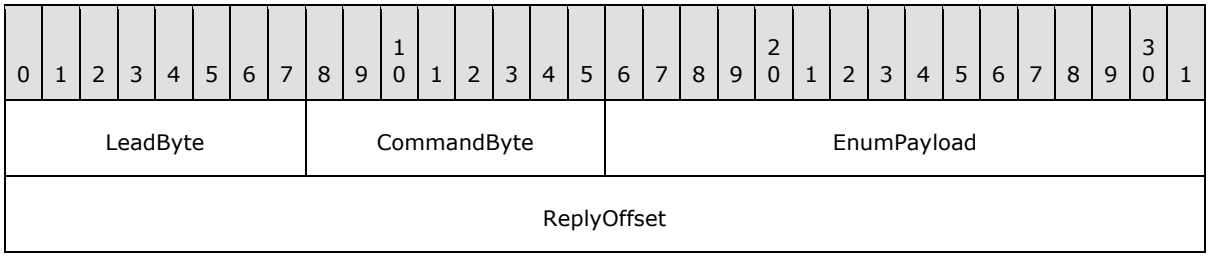
ApplicationPayload (variable): This variable-length optional field, when present, MUST be a multiple of 8 bits in length. Note that the receiver is expected to discover the size of the **ApplicationPayload** field by examining the total size of the message delivered by the underlying DirectPlay 8 service provider, because this is the only variable-length field in this message. No explicit-size field is provided. The contents of this field are application-specific, and allow the DirectPlay 8 client application to send additional application-specific information to the DirectPlay 8 server application, which it can then use to decide if it will reply to the EnumQuery, and/or determine what additional information, if any, it should return in the EnumResponse message.

2.2.2 EnumResponse

When an EnumQuery message is received by a DirectPlay 8 server application, it MAY choose to reply with an EnumResponse message. The DirectPlay 8 server application SHOULD NOT respond to any EnumQuery messages where the **QueryType** field is 0x01, and the **ApplicationGUID** field does not match the DirectPlay 8 server application's GUID.

The EnumResponse message MUST be sent to the address from which the EnumQuery message was sent. The form of this address will depend on the DirectPlay 8 Service Provider that is being used. For example, in an IPv4 service provider, the address would consist of an IPv4 style address:port pair. The response MUST be sent from the address to which the DirectPlay 8 client application MAY connect if it chooses to join the DirectPlay 8 session.

The sizes of the variable-length fields in the EnumQuery message are limited by whatever limit is placed on the overall message size by the DirectPlay 8 Service Provider that is used to transmit the message.



ResponseSize
ApplicationDescSize
ApplicationDescFlags
MaxPlayers
CurrentPlayers
SessionNameOffset
SessionNameSize
PasswordOffset
PasswordSize
ReservedDataOffset
ReservedDataSize
ApplicationReservedDataOffset
ApplicationReservedDataSize
ApplicationInstanceGUID
...
...
...
ApplicationGUID
...
...
...

SessionName (variable)
...
ReservedData
...
ApplicationReservedData (variable)
...
ApplicationData (variable)
...

LeadByte (1 byte): This field is 8 bits in length. It MUST be 0x00.

CommandByte (1 byte): This field is 8 bits in length. It MUST be 0x03.

EnumPayload (2 bytes): This field is 16 bits in length. The **EnumPayload** is a value selected by the sender of the EnumQuery message that can be used to match EnumResponse messages to their corresponding EnumQuery.

Each EnumResponse message is generated because an EnumQuery message was received. The **EnumPayload** field in the EnumResponse message MUST match the **EnumPayload** field in the EnumQuery message that generated this EnumResponse message, so the DirectPlay8 client application can track which EnumQuery message this EnumResponse message is responding to.

ReplyOffset (4 bytes): This field is 32 bits in length. This field indicates the zero based offset, in bytes, of the **ApplicationData** field within the EnumResponse message. This field will be zero if no **ApplicationData** field is contained in this message.

ResponseSize (4 bytes): This field is 32 bits in length. This field indicates the size, in bytes, of the **ApplicationData** field within the EnumResponse message. This field will be zero if no **ApplicationData** field is contained in this message.

ApplicationDescSize (4 bytes): This field is 32 bits in length. Its value MUST be 0x00000050. It represents the sum of the size of this field plus the **ApplicationDescFlags**, **MaxPlayers**, **CurrentPlayers**, **SessionNameOffset**, **SessionNameSize**, **PasswordOffset**, **PasswordSize**, **ReservedDataOffset**, **ReservedDataSize**, **ApplicationReservedDataOffset**, **ApplicationReservedDataSize**, **ApplicationInstanceGUID**, and **ApplicationGUID** fields.

ApplicationDescFlags (4 bytes): This field is 32 bits in length. It is a combination of the following bit flags:

Value	Meaning
0x00000001	If this flag is set, the DirectPlay 8 server application is hosting a client/server DirectPlay 8 session. If this flag is clear, the DirectPlay 8 server application is hosting a peer-to-peer DirectPlay 8 session.
0x00000004	If this flag is set, the DirectPlay 8 server application is hosting a DirectPlay 8 session that supports host migration. If this flag is clear, the DirectPlay 8 server application is hosting a DirectPlay 8 session that does not support host migration.
0x00000040	If this flag is set, the DirectPlay 8 server application is hosting a DirectPlay 8 session that is not enumerable via well-known port 6073. If this flag is clear, the DirectPlay 8 server application is hosting a DirectPlay 8 session that can be enumerated on well-known port 6073.
0x00000080	If this flag is set, the DirectPlay 8 server application is hosting a DirectPlay 8 session that requires a password to join. If this flag is clear, the DirectPlay 8 server application is hosting a DirectPlay 8 session that does not require a password to join.
0x00000100	If this flag is set, the DirectPlay 8 server application will not respond to enumerations. Therefore, this flag will never be set in an EnumResponse message.
0x00000200	If this flag is set, the DirectPlay 8 server application is hosting a DirectPlay 8 session that uses fast message signing. If this flag is not set, the DirectPlay 8 server application is hosting a DirectPlay 8 session that does not use fast message signing. For details about fast message signing, see [MS-DPDX].
0x00000400	If this flag is set, the DirectPlay 8 server application is hosting a DirectPlay 8 session that uses full message signing. If this flag is not set, the DirectPlay 8 server application is hosting a DirectPlay 8 session that does not use full message signing. For details about full message signing, see [MS-DPDX].

Note Flags 0x00000200 and 0x00000400 will never both be set, because a DirectPlay 8 session MUST never use both fast message signing and full message signing at the same time.

MaxPlayers (4 bytes): This field is 32 bits in length. It contains the maximum number of players that can join the DirectPlay 8 session identified by this EnumResponse message.

CurrentPlayers (4 bytes): This field is 32 bits in length. It contains the number of players currently in the DirectPlay 8 session at the time the EnumResponse message was sent by the DirectPlay 8 server application. Note that by the time the EnumResponse is received by the DirectPlay 8 client application, the number of players in the session may have changed.

SessionNameOffset (4 bytes): This field is 32 bits in length. It contains the zero-based offset, in bytes, of the session name field within this EnumResponse message. If no session name is contained in the EnumResponse message, this field will be 0.

SessionNameSize (4 bytes): This field is 32 bits in length. It contains the size, in bytes, of the session name field within this EnumResponse message. If no session name is contained in the EnumResponse message, this field will be 0.

PasswordOffset (4 bytes): This field is 32 bits in length. It is not used in the EnumResponse message and will be 0.

PasswordSize (4 bytes): This field is 32 bits in length. It is not used in the EnumResponse message and will be 0.

ReservedDataOffset (4 bytes): This field is 32 bits in length. It is not used in the EnumResponse message and will be 0. It was intended to be used for future extensions to the DirectPlay 8 Protocol, but as DirectPlay 8 is now deprecated, it will never be used.

ReservedDataSize (4 bytes): This field is 32 bits in length. It is not used in the EnumResponse message and will be 0. It was intended to be used for future extensions to the DirectPlay 8 Protocol, but as DirectPlay 8 is now deprecated, it will never be used.

ApplicationReservedDataOffset (4 bytes): This field is 32 bits in length. It contains the zero-based offset, in bytes, of the **ApplicationReservedData** field within this EnumResponse message. If no **ApplicationReservedData** is contained in the EnumResponse message, this field will be 0.

ApplicationReservedDataSize (4 bytes): This field is 32 bits in length. It contains the size, in bytes, of the **ApplicationReservedData** field within this EnumResponse message. If no **ApplicationReservedData** is contained in the EnumResponse message, this field will be 0.

ApplicationInstanceGUID (16 bytes): This field is 128 bits in length. It contains the GUID that identifies the particular instance of the DirectPlay 8 server application that generated this EnumResponse message. Each instance of a DirectPlay 8 server application generates a new GUID each time it chooses to host a new DirectPlay 8 session. Since GUIDs are by definition unique, each DirectPlay 8 session will have a unique **ApplicationInstanceGUID**. For more information about the GUID type, see [MS-DTYP] section 2.3.2.

ApplicationGUID (16 bytes): This field is 128 bits in length. It contains the GUID that identifies the DirectPlay 8 server application type that generated this EnumResponse. Each game MUST generate its own GUID to identify itself, and all DirectPlay 8 server applications for that game share the same **ApplicationGUID** which identifies the type of the DirectPlay 8 server application.

SessionName (variable): This optional field is of variable-length. Its zero-based offset within the EnumResponse message is specified by the **SessionNameOffset** field. The size, in bytes, of this field is specified by the **SessionNameSize** field. This field contains the human-readable name of the session in 16-bit Unicode characters. There is no termination character.

ReservedData (8 bytes): This field was intended to be used for future extensions to the DirectPlay 8 Protocol, but as DirectPlay 8 is now deprecated, it will never be used.

ApplicationReservedData (variable): This optional field is of variable length. Its zero-based offset within the EnumResponse message is specified in the **ApplicationDataOffset** field. Its size, in bytes, is specified in the **ApplicationDataSize** field. The contents of this field are determined by the game. This field is intended to represent game-specific data that changes infrequently. For example, data that is set when the DirectPlay 8 session is created, but does not change thereafter, is appropriate for use in this field.

ApplicationData (variable): This optional field is of variable length. Its zero-based offset within the EnumResponse message is specified in the **ReplyOffset** field. Its size, in bytes, is specified in the **ResponseSize** field. The contents of this field are determined by the game. This field is intended to represent game-specific data that changes frequently. For example, data that changes as the DirectPlay 8 session is used, such as the current state of the game, is appropriate for use in this field.

3 Protocol Details

3.1 Common 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 described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

A DirectPlay 8 server application MUST be hosting a DirectPlay 8 session to be eligible to reply to EnumQuery messages. A DirectPlay 8 server application that is hosting a DirectPlay 8 session SHOULD maintain the following DirectPlay 8 state information to be able to reply to an EnumQuery message.

- ApplicationDescFlags
- MaxPlayers
- CurrentPlayers
- SessionName (if any)
- Password (if any)

Additionally, a game MAY also maintain additional game specific state as follows:

- ApplicationReservedData
- ApplicationData

For detailed descriptions of each of these items, see the description of the EnumResponse message in section 2.2.2.

For more details on what it means to host a DirectPlay 8 session, see [MS-DPDX].

A DirectPlay 8 client application MAY send EnumQuery messages at any time to any destination it wishes. It is useful for the DirectPlay 8 client application to keep a record of the EnumQuery messages it has sent in the recent past so it can correlate any replies it receives with the original EnumQuery message. This enables the DirectPlay 8 client application to measure the round-trip time between itself and the responding DirectPlay 8 server application(s). It also enables the DirectPlay 8 client application to notice any packet loss that may be occurring between itself and any responding DirectPlay 8 server application(s).

3.1.2 Timers

EnumQuery and EnumResponse messages are delivered using the DirectPlay 8 Service Providers, which do not offer reliable message delivery. Therefore, to achieve a degree of reliability, and to enable the collection of round-trip time and packet loss data, it is useful for the DirectPlay 8 client application to send multiple EnumQuery messages spaced over a period of time. It would be appropriate to use a timer to manage the process of sending EnumQuery messages at regular intervals. The frequency of EnumQuery messages is implementation-defined. The DirectPlay 8 Protocol: Host and Port Enumeration places no restrictions on this frequency, or the number of EnumQuery messages that are sent.

3.1.3 Initialization

A DirectPlay 8 server application MUST be hosting a DirectPlay 8 session before it can respond to any EnumQuery messages.

Note that when using the IPv4 or IPv6 service provider, a DirectPlay 8 server application MAY specify which UDP port it wants to use to listen for incoming messages. The DirectPlay 8 client application MUST be aware of this port selection in order to direct the EnumQuery message to the correct port. UDP port 6073 has been registered with IANA for use by DirectPlay 8. If the DirectPlay 8 application has no compelling reason to use a different port, this is a good port to choose on an IPv4 or IPv6 network. Because this port is registered with IANA (as specified in [IANAPORT]), and is used by multiple games, it increases the likelihood that some firewalls may be preconfigured to allow traffic on this port. However, a game can use any port it deems appropriate, according to the rules and customs of the IP networks that it is using.

UDP port 47624 has also been registered by Microsoft with IANA for use by "Direct Play Server" as specified in [IANAPORT]. On Windows operating systems, a DirectPlay 8 server application has the option to register with a DirectPlay Server Windows component that listens for EnumQuery requests on UDP port 47624. When the DirectPlay 8 server application registers, it informs the DirectPlay Server Windows component of its ApplicationGUID. This enables the DirectPlay Server to forward any EnumQuery requests that are directed at port 47624 to any DirectPlay 8 server application that has a matching ApplicationGUID. This can be useful for running multiple instances of the same type of DirectPlay 8 server application on a single machine, because a single EnumQuery request can be delivered to all instances of the DirectPlay 8 server application that have registered with the DirectPlay Server component with a matching ApplicationGUID. Also, the DirectPlay 8 client application does not need to know which UDP ports each DirectPlay 8 server application is listening on. The DirectPlay Client application can simply direct the EnumQuery message to UDP port 47624.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

When a DirectPlay 8 server application is hosting a DirectPlay 8 session, and it receives an EnumQuery message, it MAY respond to the address from which the EnumQuery message originated with an EnumResponse message. Note that the DirectPlay 8 server application MAY choose not to reply to any particular EnumQuery message for application-specific reasons, such as server load, current game state, or any other reason.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

4 Protocol Examples

The following diagram shows an example use of the DirectPlay 8 Protocol: Host and Port Enumeration.

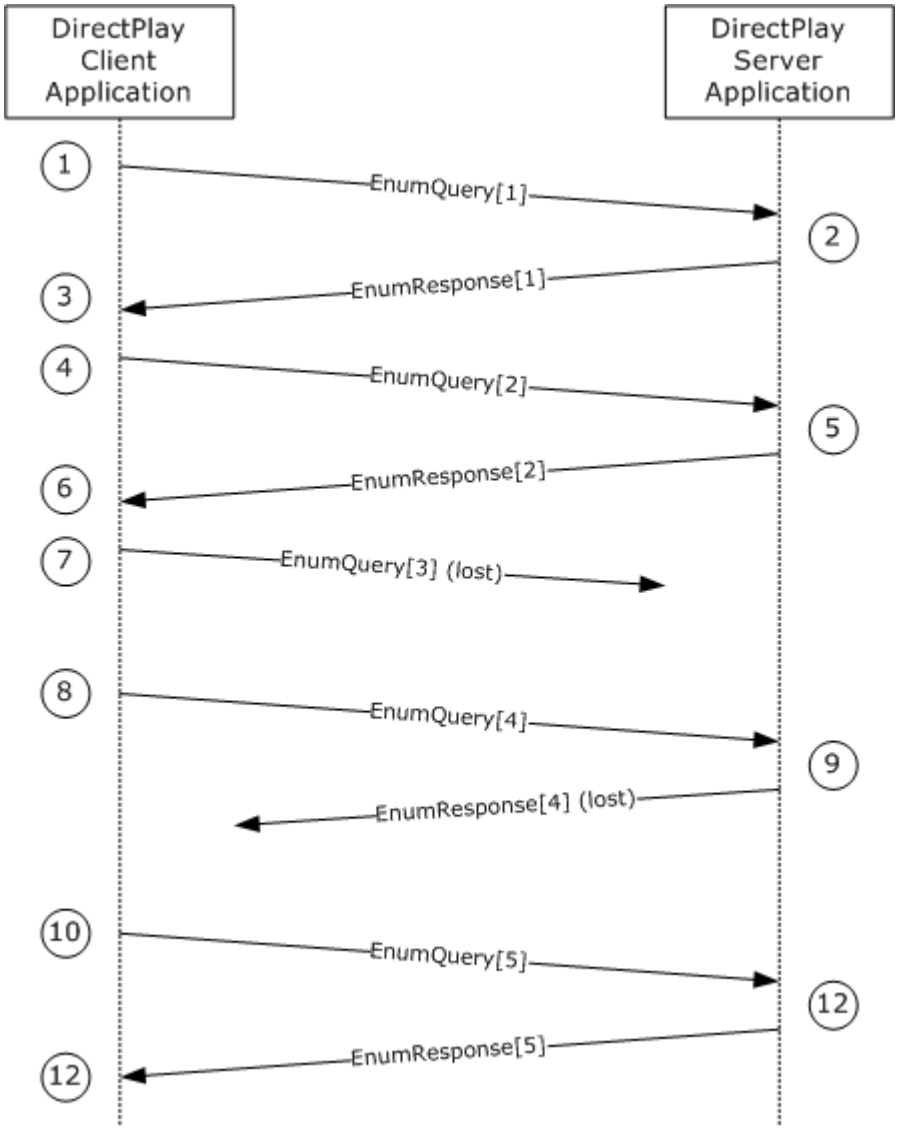


Figure 1: DirectPlay 8 Protocol: Host and Port Enumeration

The steps depicted in the diagram example are as follows:

1. The DirectPlay 8 client application sends an EnumQuery message to the DirectPlay 8 server application. This EnumQuery message contains an **EnumPayload** value of 1. The EnumQuery message is sent to the DirectPlay 8 server application directly via the selected DirectPlay 8 service provider, which does not offer reliable message delivery. Therefore, the EnumQuery message is at risk of being lost in transit. In this example step, the EnumQuery message is successfully received by the DirectPlay 8 server application.

2. The DirectPlay 8 server application receives the EnumQuery message. The DirectPlay 8 server application is hosting a DirectPlay 8 session. Based on the content of the EnumQuery message and its own internal state, it chooses to respond to the EnumQuery message with an EnumResponse message. It copies the **EnumPayload** value of 1 from the EnumQuery message to the EnumResponse message and sends the EnumResponse message back to the address that the EnumQuery message came from. The EnumResponse message is sent directly via the selected DirectPlay 8 Service Provider, which does not offer reliable message delivery. Therefore, the EnumResponse message is at risk of being lost in transit. In this example step, the EnumResponse message is successfully received by the DirectPlay 8 client application.

3. The DirectPlay 8 client application receives the EnumResponse message. Based on the content of EnumResponse message, the DirectPlay 8 client application has the information it requires to connect to the DirectPlay 8 session that is being hosted by the responding DirectPlay 8 server application. By measuring the elapsed time between sending the EnumQuery message with **EnumPayload** of 1, and receiving the EnumResponse message with **EnumPayload** of 1, the DirectPlay 8 client application can also estimate the round-trip message latency between itself and the responding DirectPlay 8 server application. In this example, the DirectPlay 8 client application chooses not to connect immediately to the DirectPlay 8 session identified in the EnumResponse message, but instead chooses to continue sending EnumQuery messages at regular intervals to the DirectPlay 8 server application.

4. After some reasonable time period following step 1, the DirectPlay 8 client application chooses to send another EnumQuery message to the DirectPlay 8 server application. This EnumQuery message contains an **EnumPayload** value of 2. The EnumQuery message is sent to the DirectPlay 8 server application directly via the selected DirectPlay 8 service provider, which does not offer reliable message delivery. Therefore, the EnumQuery message is at risk of being lost in transit. In this example step, the EnumQuery message is successfully received by the DirectPlay 8 server application.

5. The DirectPlay 8 server application receives the EnumQuery message. The DirectPlay 8 server application is hosting a DirectPlay 8 session. Based on the content of the EnumQuery message and its own internal state, it chooses to respond to the EnumQuery message with an EnumResponse message. It copies the **EnumPayload** value of 2 from the EnumQuery message to the EnumResponse message and sends the EnumResponse message back to the address that the EnumQuery message came from. The EnumResponse message is sent directly via the selected DirectPlay 8 service provider, which does not offer reliable message delivery. Therefore the EnumResponse message is at risk of being lost in transit. In this example step, the EnumResponse message is successfully received by the DirectPlay 8 client application.

6. The DirectPlay 8 client application receives the EnumResponse message. Based on the content of EnumResponse message, the DirectPlay 8 client application has the information it requires to connect to the DirectPlay 8 session that is being hosted by the responding DirectPlay 8 server application. By measuring the elapsed time between sending the EnumQuery message with **EnumPayload** of 2, and receiving the EnumResponse message with **EnumPayload** of 2, the DirectPlay 8 client application can also estimate the round trip message latency between itself and the responding DirectPlay 8 server application. The DirectPlay 8 client application now has two measurements of this round trip message latency, and therefore can make a more accurate prediction of future message latency than it could after receiving only one EnumResponse message from this DirectPlay 8 server application. This is one of the benefits of sending multiple EnumQuery messages to the same DirectPlay 8 server application. In this example, the DirectPlay 8 client application chooses to not immediately connect to the DirectPlay 8 session identified in the EnumResponse message, but instead chooses to continue sending EnumQuery messages at regular intervals to the DirectPlay 8 server application.

7. After some reasonable time period following step 4, the DirectPlay 8 client application chooses to send another EnumQuery message to the DirectPlay 8 server application. This EnumQuery

message contains an **EnumPayload** value of 3. The EnumQuery message is sent to the DirectPlay 8 server application directly via the selected DirectPlay 8 service provider, which does not offer reliable message delivery. Therefore the EnumQuery message is at risk of being lost in transit. In this example step, the EnumQuery message is lost in transit and is not received by the DirectPlay 8 server application.

8. After some reasonable time period following step 7, the DirectPlay 8 client application chooses to send another EnumQuery message to the DirectPlay 8 server application. This EnumQuery message contains an **EnumPayload** value of 4. The EnumQuery message is sent to the DirectPlay 8 server application directly via the selected DirectPlay 8 Service Provider, which does not offer reliable message delivery. Therefore, the EnumQuery message is at risk of being lost in transit. In this example step, the EnumQuery message is successfully received by the DirectPlay 8 server application.

9. The DirectPlay 8 server application receives the EnumQuery message. The DirectPlay 8 server application is hosting a DirectPlay 8 session. Based on the content of the EnumQuery message and its own internal state, it chooses to respond to the EnumQuery message with an EnumResponse message. It copies the **EnumPayload** value of 4 from the EnumQuery message to the EnumResponse message and sends the EnumResponse message back to the address that the EnumQuery message came from. The EnumResponse message is sent directly via the selected DirectPlay 8 Service Provider, which does not offer reliable message delivery. Therefore, the EnumResponse message is at risk of being lost in transit. In this example step, the EnumResponse message is lost in transit and is not received by the DirectPlay 8 client application.

10. After some reasonable time period following step 8, the DirectPlay 8 client application chooses to send another EnumQuery message to the DirectPlay 8 server application. This EnumQuery message contains an **EnumPayload** value of five. The EnumQuery message is sent to the DirectPlay 8 server application directly via the selected DirectPlay 8 service provider, which does not offer reliable message delivery. Therefore, the EnumQuery message is at risk of being lost in transit. In this example step, the EnumQuery message is successfully received by the DirectPlay 8 server application.

11. The DirectPlay 8 server application receives the EnumQuery message. The DirectPlay 8 server application is hosting a DirectPlay 8 session. Based on the content of the EnumQuery message and its own internal state, it chooses to respond to the EnumQuery message with an EnumResponse message. It copies the **EnumPayload** value of 5 from the EnumQuery message to the EnumResponse message and sends the EnumResponse message back to the address that the EnumQuery message came from. The EnumResponse message is sent directly via the selected DirectPlay 8 service provider, which does not offer reliable message delivery. Therefore, the EnumResponse message is at risk of being lost in transit. In this example step, the EnumResponse message is successfully received by the DirectPlay 8 client application.

12. The DirectPlay 8 client application receives the EnumResponse message. Based on the content of EnumResponse message, the DirectPlay 8 client application has the information it requires to connect to the DirectPlay 8 session that is being hosted by the responding DirectPlay 8 server application. By measuring the elapsed time between sending the EnumQuery message with **EnumPayload** of 5, and receiving the EnumResponse message with **EnumPayload** of 5, the DirectPlay 8 client application can also estimate the round trip message latency between itself and the responding DirectPlay 8 server application. The DirectPlay 8 client application now has three measurements of this round trip message latency, and therefore can make a more accurate prediction of future message latency than it could after receiving only two EnumResponse messages from this DirectPlay 8 server application.

This is one of the benefits of sending multiple EnumQuery messages to the same DirectPlay 8 server application. Depending on the time that has elapsed since sending the EnumQuery messages with

EnumPayload 3 and **EnumPayload** 4, the DirectPlay 8 client application may also reasonably conclude that these EnumQuery messages, or the EnumResponse messages they may have generated, have been lost in transit. With that information, the DirectPlay 8 client application can also generate an estimate of the possible future message delivery reliability. At this time, the DirectPlay 8 client application now has a reasonable estimate of the future round trip message latency and reliability, and can decide to not connect to the DirectPlay 8 session identified in the EnumResponse messages, attempt to connect to the DirectPlay 8 session identified in the EnumResponse messages, or continue to send additional periodic EnumQuery messages to obtain more information regarding the message latency and reliability between itself and the DirectPlay 8 server application.

The DirectPlay 8 client application may also have been sending EnumQuery messages to other DirectPlay 8 server applications in parallel, and may find one of those other DirectPlay 8 sessions is better in some application-specific way. The point at which a DirectPlay 8 client application chooses to stop sending EnumQuery messages to a particular DirectPlay 8 server application is application-specific. The method that a DirectPlay 8 client application uses to choose which DirectPlay 8 session to attempt to join is application-specific.

5 Security

5.1 Security Considerations for Implementers

There are no security considerations for implementers of this protocol.

5.2 Index of Security Parameters

There are no security parameters for this protocol.

6 Appendix A: Windows Behavior

The information in this specification is applicable to the following versions of Windows:

- Windows XP
- Windows Server 2003
- Windows Vista

Exceptions, if any, are noted below. Unless otherwise specified, any statement of optional behavior in this specification prescribed using the terms SHOULD or SHOULD NOT implies Windows Behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies Windows does not follow the prescription.

7 Index