# [MS-MICE-Diff]:

# Miracast over Infrastructure Connection Establishment 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 |
|---|---|---|---|
| 3/16/2017 | 1.0 | New | Released new document. |
| 6/1/2017 | 1.1 | Minor | Clarified the meaning of the technical content. |
| 3/16/2018 | 2.0 | Major | Significantly changed the technical content. |

# Table of Contents

# 1 Introduction

The Miracast over Infrastructure Connection Establishment ~~protocol~~Protocol specifies a connection negotiation sequence that is used to connect and disconnect from a Miracast over Infrastructure ~~endpoint~~device.

This protocol ~~also defines the Miracast over Infrastructure~~uses a Wi-Fi Simple Configuration (WSC) information element (IE) ~~vendor extension~~Vendor Extension attribute~~, which helps identify Miracast receivers (sinks~~ to advertise a receiver (Sink) that can support Miracast ~~sessions~~ over infrastructure ~~links in addition to Wi-Fi Direct (WFD) links~~sessions.

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:

**802.11 Access Point (AP)**: Any entity that has IEEE 802.11 functionality and provides access to the distribution services, via the wireless medium for associated stations (STAs).

**ASCII**: The American Standard Code for Information Interchange (ASCII) is an 8-bit character-encoding scheme based on the English alphabet. ASCII codes represent text in computers, communications equipment, and other devices that work with text. ASCII refers to a single 8-bit ASCII character or an array of 8-bit ASCII characters with the high bit of each character set to zero.

**basic service set identifier (BSSID)**: A 48-bit structure that is used to identify an entity such as the access point in a wireless network. This is typically a MAC address.

**Beacon**: A management frame that contains all of the information required to connect to a network. In a WLAN, Beacon frames are periodically transmitted to announce the presence of the network.

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

**Domain Name System (DNS)**: A hierarchical, distributed database that contains mappings of domain names to various types of data, such as IP addresses. DNS enables the location of computers and services by user-friendly names, and it also enables the discovery of other information stored in the database.

**friendly name**: A name for a user or object that can be read and understood easily by a human.

**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).

**information element (IE)**: In a Wi-Fi Protected Setup (WPS) scenario, descriptive information consisting of informative type-length-values that specify the possible and currently deployed configuration methods for a device. The IE is transferred and added to the Beacon and Probe Response frames, and optionally to the Probe Request frame and associated request and response messages.

**Internet Protocol version 4 (IPv4)**: An Internet protocol that has 32-bit source and destination addresses. IPv4 is the predecessor of IPv6.

**Internet Protocol version 6 (IPv6)**: A revised version of the Internet Protocol (IP) designed to address growth on the Internet. Improvements include a 128-bit IP address size, expanded routing capabilities, and support for authentication and privacy.

**organizationally unique identifier (OUI)**: A unique 24-bit string that uniquely identifies a vendor, manufacturer, or organization on a worldwide l basis, as specified in [IEEE-OUI]. The OUI is used to help distinguish both physical devices and software, such as a network protocol, that belong to one entity from those that belong to another.

**peer –to –peer (P2P)**: An Internet-based networking option in which two or more computers connect directly to each other ~~in order~~ to communicate and share files without use of a central server.

**Probe Request**: A frame that contains the advertisement IE for a device that is seeking to establish a connection with a proximate device. The Probe Request frame is defined in the Wi-Fi Peer-to-Peer (P2P) Specification v1.2 [WF-P2P1.2] section 4.2.2.

**Probe Response**: A frame that contains the advertisement IE for a device. The Probe Response is sent in response to a Probe Request. The Probe Response frame is defined in the Wi-Fi Peer-to-Peer (P2P) Specification v1.2 [WF-P2P1.2] section 4.2.3.

**Real-Time Streaming Protocol (RTSP)**: A protocol used for transferring real-time multimedia data (for example, audio and video) between a server and a client, as specified in [RFC2326]. It is a streaming protocol; this means that RTSP attempts to facilitate scenarios in which the multimedia data is being simultaneously transferred and rendered (that is, video is displayed and audio is played).

**Stock Keeping Unit (SKU)**: A unique code that refers to a particular manufactured object or source of revenue. A SKU can refer to a retail product (software in a box that is sold through a channel), a subscription program (such as MSDN), or an online service (such as MSN).

**subnet**: A logical division of a network. Subnets provide a multilevel hierarchical routing structure for the Internet. On TCP/IP networks, subnets are defined as all devices whose IP addresses have the same prefix. Subnets are useful for both security and performance reasons. In general, broadcast messages are scoped to within a single subnet. For more information about subnets, see [RFC1812].

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

**type-length-value (TLV)**: A property of a network interface, so named because each property is composed of a Type field, a Length field, and a value.

~~**User Datagram Protocol (UDP)**: The connectionless protocol within TCP/IP that corresponds to the transport layer in the ISO/OSI reference model.~~

**UTF-16**: A standard for encoding Unicode characters, defined in the Unicode standard, in which the most commonly used characters are defined as double-byte characters. Unless specified otherwise, this term refers to the UTF-16 encoding form specified in [UNICODE5.0.0/2007] section 3.9.

~~**UTF-8**: A byte-oriented standard for encoding Unicode characters, defined in the Unicode standard. Unless specified otherwise, this term refers to the UTF-8 encoding form specified in [UNICODE5.0.0/2007] section 3.9.~~

**virtual private network (VPN)**: A network that provides secure access to a private network over public infrastructure.

**Wi-Fi Direct (WFD)**: A standard that allows Wi-Fi devices to connect to each other without requiring a wireless access point (WAP). This standard enables WFD devices to transfer data directly among each other resulting in significant reductions in setup.

**Wi-Fi Protected Setup (WPS)**: A computing standard that attempts to allow easy establishment of a secure wireless home network. This standard was formerly known as Wi-Fi Simple Config.

**wireless access point (WAP)**: A wireless network access server (NAS) that implements 802.11.

**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.

[IANA-DNS] IANA, "Domain Name System (DNS) Parameters", April 2009, http://www.iana.org/assignments/dns-parameters

[IANAPORT] IANA, "Service Name and Transport Protocol Port Number Registry", http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

~~[IANA] IANA, "Internet Assigned Numbers Authority (IANA)", http://www.iana.org~~

[IEEE802.11-2012] IEEE, "Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", ANSI/IEEE Std 802.11-2012, http://standards.ieee.org/getieee802/download/802.11-2012.pdf

**Note** There is a charge to download this document.

[RFC1034] Mockapetris, P., "Domain Names - Concepts and Facilities", STD 13, RFC 1034, November 1987, http://www.ietf.org/rfc/rfc1034.txt

[RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", RFC 1123, October 1989, http://www.ietf.org/rfc/rfc1123.txt

[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

[RFC2181] Elz, R., and Bush, R., "Clarifications to the DNS Specification", RFC 2181, July 1997, http://www.ietf.org/rfc/rfc2181.txt

[RFC2326] Schulzrinne, H., Rao, A., and Lanphier, R., "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998, http://www.ietf.org/rfc/rfc2326.txt

[RFC4291] Hinden, R. and Deering, S., "IP Version 6 Addressing Architecture", RFC 4291, February 2006, http://www.ietf.org/rfc/rfc4291.txt

[RFC6762] Krochmal, M. and Cheshire, S., "Multicast DNS", http://www.rfc-editor.org/rfc/rfc6762.txt

[~~RFC768] Postel, J., "User Datagram Protocol", STD 6,~~RFC6763] Cheshire, S. and Krochmal, M., "DNS-Based Service Discovery", RFC ~~768, August 1980, http~~2326, April 1998, https://www.rfc-editor.org/rfc/~~rfc768~~rfc6763.txt

[RFC793] Postel, J., Ed., "Transmission Control Protocol: DARPA Internet Program Protocol Specification", RFC 793, September 1981, http://www.rfc-editor.org/rfc/rfc793.txt

[WF-DTS1.1] Wi-Fi Alliance, "Wi-Fi Display Technical Specification v1.1", April 2014, https://www.wi-fi.org/file/wi-fi-display-technical-specification-v11

**Note** There is a charge to download the specification.

[WF-P2P1.2] Wi-Fi Alliance, "Wi-Fi Peer-to-Peer (P2P) Technical Specification v1.2", https://www.wi-fi.org/wi-fi-peer-to-peer-p2p-technical-specification-v12

**Note** There is a charge to download the specification.

[WF-WSC2.0.2] Wi-Fi Alliance, "Wi-Fi Simple Configuration Technical Specification v2.0.2", August 2011, https://www.wi-fi.org/wi-fi-simple-configuration-technical-specification-v202

**Note** There is a charge to download the specification.

### 1.2.2 Informative References

[IEEE-OUI] IEEE Standards Association, "IEEE OUI Registration Authority", February 2007, http://standards.ieee.org/regauth/oui/oui.txt

[WF-DTS2.1] Wi-Fi Alliance, "Wi-Fi Display Technical Specification v2.0", April 2016, https://www.wi-fi.org/file/wi-fi-display-technical-specification-v21

## 1.3 Overview

The Miracast over Infrastructure protocol ~~defines~~provides the ability to transmit a multimedia data stream over a local wireless network instead of Wi-Fi Direct (WFD). The process of such transmission is called "projection".

A Miracast over Infrastructure session involves the following ~~actors:~~principals.

**Miracast Source:** ~~The~~A device that sends audio and video streams to the Miracast Sink. This device is sometimes called a "sender". Optionally, this device can also receive input signals from the Miracast Sink.

**Miracast Sink:** ~~The~~A device that receives audio and video streams from the Miracast Source. This device is sometimes called a "receiver". Optionally, this device can also send input signals back to the Miracast Source.

The following diagram ~~shows~~illustrates the ~~lifelines~~logical flow of ~~actors in~~establishing a Miracast over Infrastructure session~~.~~, including successful and unsuccessful outcomes. For further details, see Protocol Details (section 3).

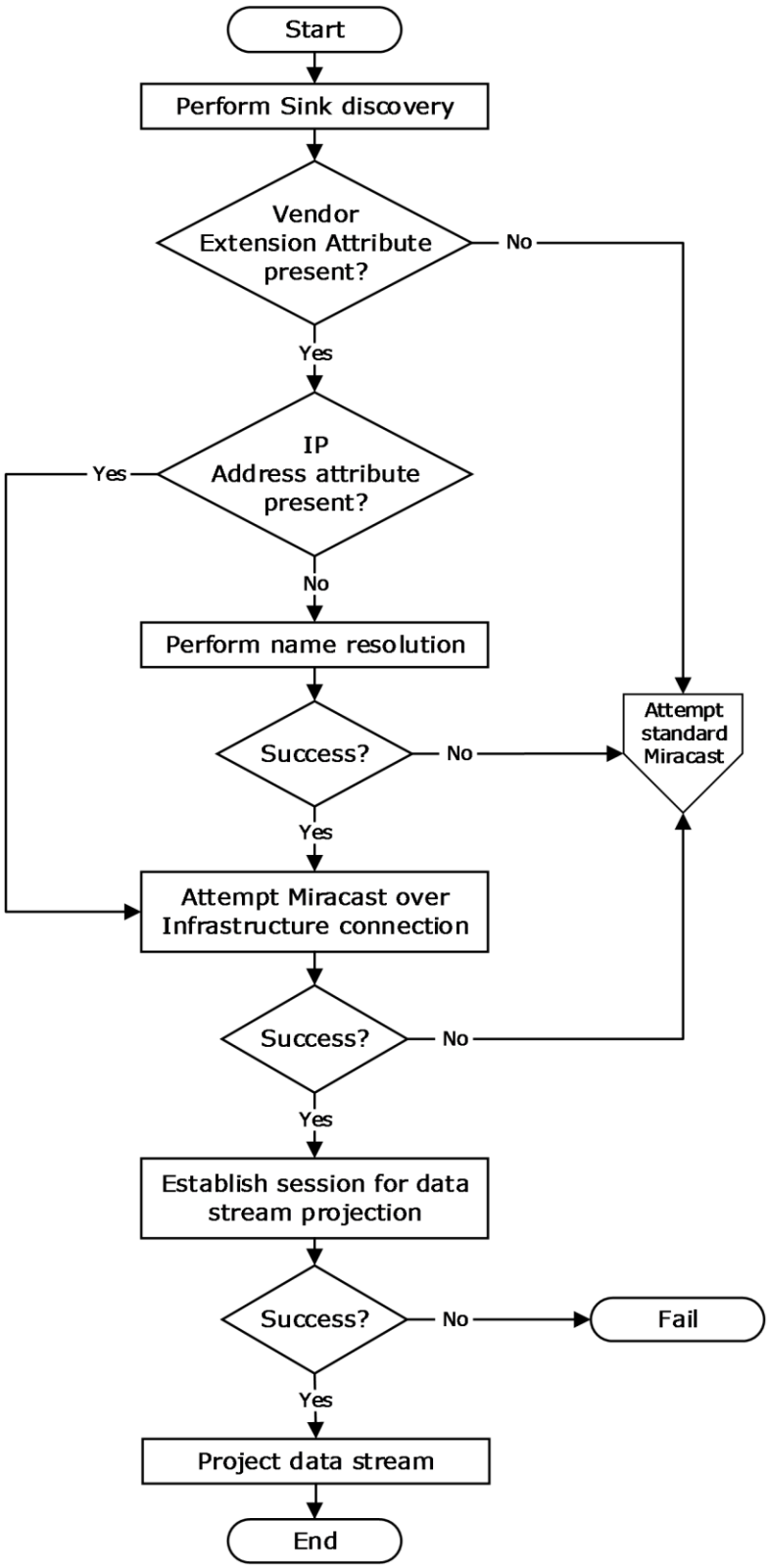



**Figure** 1**: A Miracast over Infrastructure ~~actor lifelines~~**

~~A~~ **session** ~~between a Miracast Source and Sink is established before projection can start. Sessions include the following phases.~~

1. ~~Wi-Fi Direct (WFD) device discovery ([WF-DTS1.1] section 4.9.2).~~

   ~~Management frames are exchanged, including Beacons, Probe Requests, and Probe Responses ([WF-P2P1.2] section 4.2). The Wi-Fi Simple Configuration (WSC) information element (IE) vendor extension attribute (section 2.2.3) is used to advertise the presence of devices that can perform WSC operations.~~

2. ~~The Source resolves the host name of the target Sink device by initiating DNS and/or Multicast DNS (mDNS) [RFC6762] queries.~~

3. ~~Source messages (section 2.2.1) to communicate to the Sink about starting and stopping the projection.~~

~~Implementing the Miracast over Infrastructure protocol requires an understanding of session establishment over standard Miracast.~~

~~First, the connection to a Sink over standard Miracast is established over Wi-Fi Direct (WFD). Because Miracast over Infrastructure does not require WFD, a way is needed to inform the Miracast Source that a Miracast over Infrastructure connection is possible, even if the target Sink isn't on the same subnet as the Source. That ability is provided through a WSC IE vendor extension attribute (section 2.2.3).~~

~~Second, standard Miracast is triggered to start automatically when a WFD session is established. Because the Miracast over Infrastructure protocol is not triggered by WFD, messages have been defined for starting and stopping Miracast over Infrastructure sessions (section 2.2.1).~~

~~WFD device discovery is performed, even if the session connection might later be made by using the Miracast over Infrastructure protocol. If the connection cannot be made by using this protocol, the Source falls back to a standard WFD Miracast session.~~

~~The following diagram illustrates the attempt to establish a Miracast over Infrastructure session, along with some possible outcomes.~~

~~**Figure : Establishing a Miracast over Infrastructure session**~~

~~When a Source is ready to project to a Sink, it listens on its control port for Real-Time Streaming Protocol (RTSP) (7236 by default) for connection requests and then sends a Source Ready message to the Sink. The Sink is expected to be listening for Source Ready messages on TCP port 7250, and the Source connects to port 7250 to deliver RTSP port information to the Sink in the Source Ready message (section 2.2.1.1). In turn, the Sink connects to the specified RTSP Source port to establish the link.~~

~~To pause or stop the projection, the Source sends a Stop Projection message (section 2.2.1.2) to notify the Sink. Upon receipt of that message, the Sink stops displaying the stream, and a disconnection follows from the Source on the socket that is connected on port 7250.~~

A Miracast over Infrastructure session consists of three phases: device discovery, host name resolution, and projection.

The device discovery phase starts with a Miracast Source trying to find devices capable of performing the role of Miracast Sink. Each Sink advertises its capabilities by transmitting Beacon and Probe Response frames that might include WSC IE Vendor Extension attributes (section 2.2.4).

A Sink is selected from those discovered; for example, by asking a user to pick one. If the Vendor Extension attribute from the selected Sink does not indicate support for Miracast over Infrastructure, the Source falls back to using standard Miracast [WF-WSC2.0.2].

If the Vendor Extension attribute contains one or more IP Address attributes (section 2.2.4.5), the Source optionally skips the host name resolution phase and proceeds to the projection phase.

In the host name resolution phase, name resolution is performed on the name in a Host Name attribute (section 2.2.4.2) in the Vendor Extension attribute. If name resolution is unsuccessful, the Source again falls back to using standard Miracast.

In the projection phase, the Source attempts a connection to the Sink for sending Miracast over Infrastructure messages (section 2.2). Finally, the Sink establishes a connection with the Source for streaming multimedia data. If that connection cannot be established, the entire process fails.

## 1.4 Relationship to Other Protocols

The Miracast over Infrastructure protocol builds upon the following standard technologies~~:~~ .

- Domain Name System (DNS) [IANA-DNS] [RFC1034] [RFC2181]
- Multicast DNS (mDNS) [RFC6762]
- Real-Time Streaming Protocol (RTSP) [RFC2326]
- Transmission Control Protocol (TCP) [RFC793]
- ~~User Datagram Protocol [RFC768]~~
- Wi-Fi Display Protocol [WF-DTS1.1]
- Wi-Fi Peer-to-Peer (P2P) Protocol [WF-P2P1.2]
- Wi-Fi Simple Configuration (WSC) Protocol [WF-WSC2.0.2]

## 1.5 Prerequisites/Preconditions

~~The~~ Miracast over Infrastructure ~~protocol requires that~~has the following ~~both be true:~~prerequisites and preconditions.

- The Miracast Source and Miracast Sink endpoints are on the same logical IP network, so they can establish a ~~Miracast over Infrastructure~~local network connection.
- Either the Sink is on the same logical IP subnet as the Source, or the Sink's name is registered in a DNS server that the Source can resolve to.

## 1.6 Applicability Statement

~~The~~ Miracast over Infrastructure ~~protocol~~ is applicable to ~~projecting~~streaming audio and video content from one device to another, such as PC to large-screen TV, PC to PC, phone to PC, and so on.

The protocol functions in a configuration in which Miracast Source and Miracast Sink devices share a common logical IP network and determine they can project content across that network.

## 1.7 Versioning and Capability Negotiation

~~This is the first version of the protocol.~~

Versioning and capability negotiation are performed by using Vendor Extension attributes (section 2.2.4).

## 1.8 Vendor-Extensible Fields

None.

## 1.9 Standards Assignments

The Miracast over Infrastructure protocol uses the following standard port assignments.

| Parameter | Value | Reference |
|---|---|---|
| ~~Multicast DNS~~ | ~~5353~~ | ~~[IANAPORT]~~ |
| ~~RTSP requests (both UDP and TCP)~~TCP port | ~~7236~~7250 | [IANAPORT] |
| ~~TCP~~ | ~~7250~~ | ~~[IANA]~~ |

# 2 Messages

## 2.1 Transport

~~The~~ Miracast over Infrastructure ~~Source Ready and Stop Projection~~ messages (section 2.2~~.1~~) are sent over TCP port 7250~~.~~

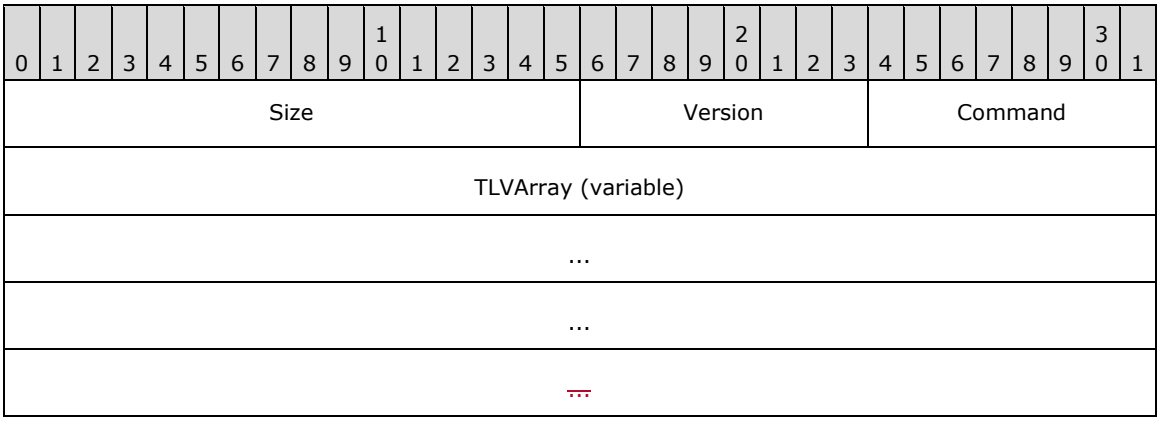
~~The Multicast DNS (mDNS) [RFC6762] messages utilize~~ to manage the ~~standard UDP transport [RFC768] on port 5353.~~multimedia stream.

## 2.2 Message Syntax

In the structures defined in this section, multi-byte field values are ordered in big-endian format, unless specified otherwise, and string values do not include NUL terminators.

### ~~2.2.1 Miracast Messages~~

This section defines the messages used for starting and stopping Miracast over Infrastructure sessions. This is the general format for Miracast messages~~:~~.




**Size (2 bytes):** The size of the message, in bytes.

**Version (1 byte):** The version of this protocol, which is 0x01.

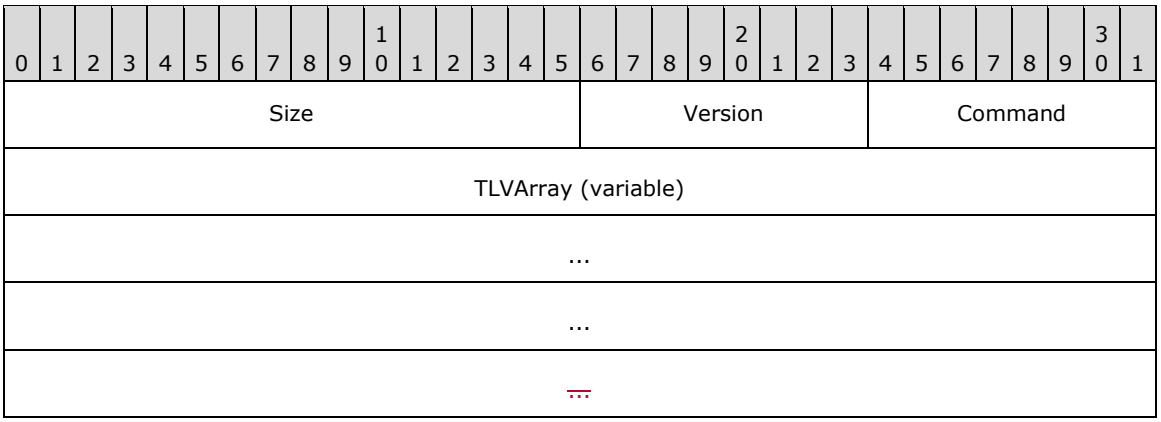
**Command (1 byte):** The type of message, which determines the TLVs passed in the **TLVArray** field. The following messages are defined in the sections listed.

| Message type | Section | Description |
|---|---|---|
| SOURCE_READY<br>0x01 | 2.2.1~~.1~~ | Indicates the Miracast Source is ready to accept a connection on the RTSP port. |
| STOP_PROJECTION<br>0x02 | 2.2.~~1.~~2 | Indicates the end of the projection. |

**TLVArray (variable):** An array of one or more Miracast TLVs (section 2.2.~~1.~~3), which specify information for the message.

### ~~2.2.2~~2.2.1 Source Ready Message

The Source Ready message is sent by the Miracast Source to the Miracast Sink when the Source has started listening on the RTSP port and is ready to accept an incoming connection on it.




**Size (2 bytes):** The size of the entire message, in bytes.

**Version (1 byte):** The version of this protocol, which is 0x01.

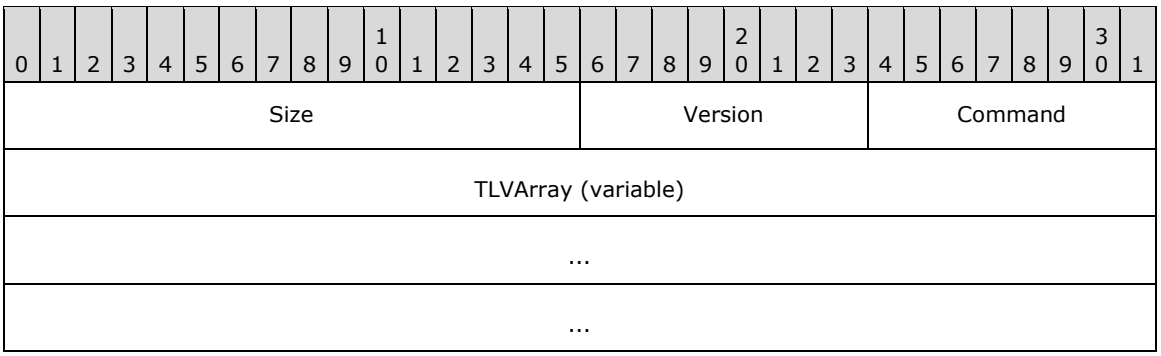
**Command (1 byte):** The type of message, which is 0x01 for **SOURCE_READY**.

**TLVArray (variable):** The following TLVs, included in any order:

- Friendly Name TLV (section 2.2.~~1.~~3.1)
- RTSP Port TLV (section 2.2.~~1.~~3.2)
- Source ID TLV (section 2.2.~~1.~~3.3)

### ~~2.2.3~~2.2.2 Stop Projection Message

The Stop Projection message is sent by the Miracast Source to notify the Miracast Sink that the projection is being stopped.




**Size (2 bytes):** The size of the entire message, in bytes.

**Version (1 byte):** The version of this protocol, which is 0x01.

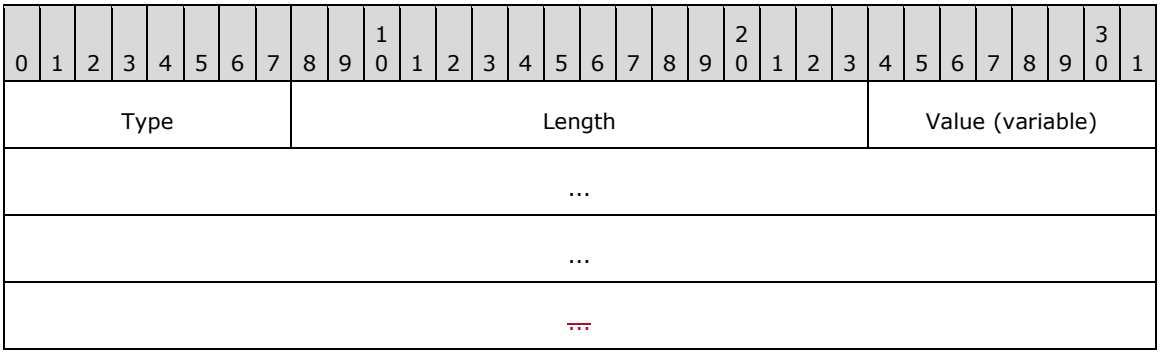
**Command (1 byte):** The type of message, which is 0x02 for **STOP_PROJECTION**.

**TLVArray (variable):** The following ~~structures~~TLVs, included in any order:

- Friendly Name TLV (section 2.2.~~1.~~3.1)
- Source ID TLV (section 2.2.~~1.~~3.3)

### ~~2.2.4~~2.2.3 Miracast TLVs

This section defines common type-length-value (TLV) structures that are used to pass information in messages during a Miracast session. This is the general format for the TLVs:




**Type (1 byte):** The type of TLV, which determines the information passed in the **Value** field. The following TLVs are defined in the sections listed.

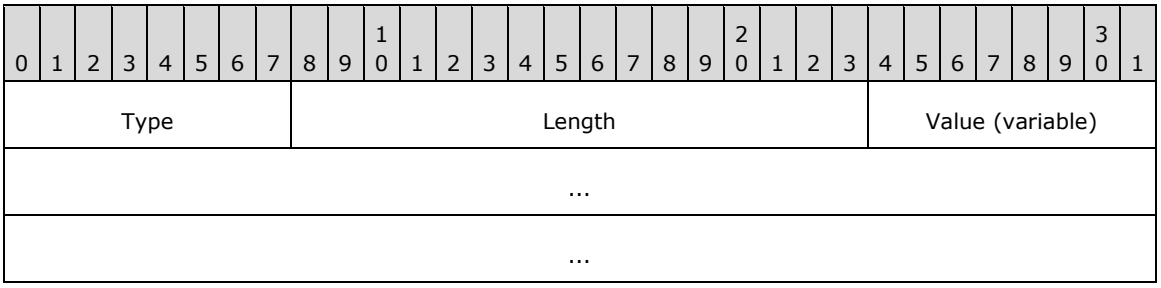
| TLV type | Section | Description |
|---|---|---|
| FRIENDLY_NAME<br>0x00 | 2.2.~~1.~~3.1 | Specifies the friendly name of the Miracast Source. |
| RTSP_PORT<br>0x02 | 2.2.~~1.~~3.2 | Specifies the port on which the Source is listening for RTSP connections. |
| SOURCE_ID<br>0x03 | 2.2.~~1.~~3.3 | Specifies an identifier for the Source, which is used for all messages sent during a Miracast session. |

**Length (2 bytes):** The length of the **Value** field, in bytes. This value MUST be greater than or equal to 0x0001.

**Value (variable):** One or more bytes, which specify information for the TLV.

#### ~~2.2.4.1~~2.2.3.1 Friendly Name TLV

The Friendly Name TLV specifies the friendly name of the Miracast Source in messages to the Miracast Sink.



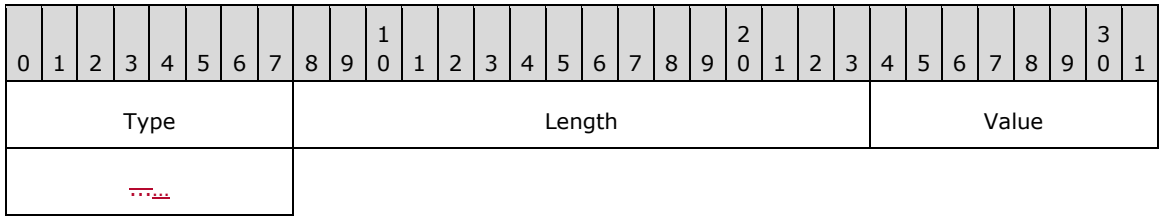

**Type (1 byte):** The type of TLV, which is 0x00 for the Friendly Name TLV.

**Length (2 bytes):** The length of the **Value** field, in bytes.

**Value (variable):** The friendly name string of the Source, encoded in UTF-16.

### ~~2.2.4.2~~2.2.3.2 RTSP Port TLV

The RTSP Port TLV specifies the port on which the Miracast Source is listening. The port is used in messages for connecting sessions over RTSP.



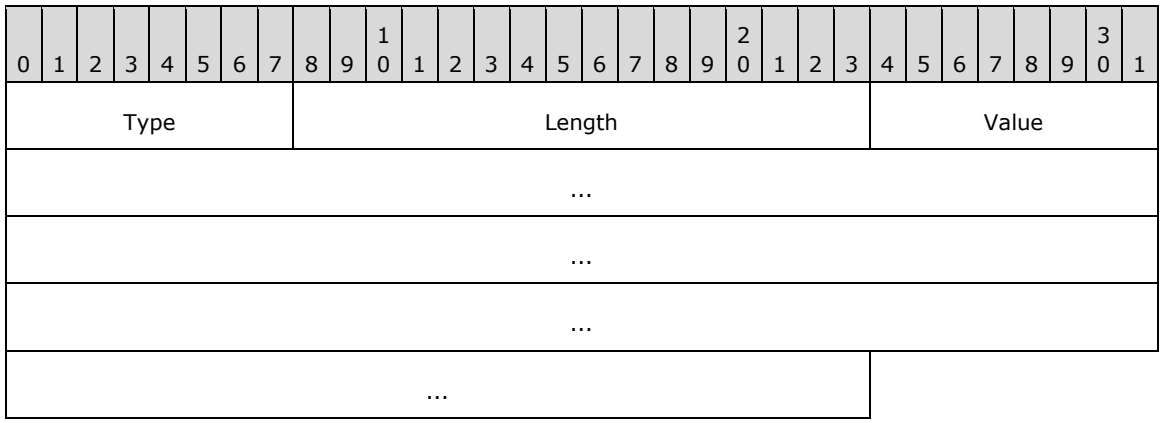

**Type (1 byte):** The type of TLV, which is 0x02 for the RTSP Port TLV.

**Length (2 bytes):** The length of the **Value** field, in bytes, which is 0x0002.

**Value (2 bytes):** The RTSP port on which the Source is listening (7236 by default).

### ~~2.2.4.3~~2.2.3.3 Source ID TLV

The Source ID TLV specifies a unique identifier for the Miracast Source. That identifier is used in all messages sent during a session.




**Type (1 byte):** The type of TLV, which is 0x03 for the Source ID TLV.

**Length (2 bytes):** The length of the **Value** field, in bytes, which is 0x0010.

**Value (16 bytes):** An implementation-defined value that identifies the Source.

### ~~2.2.5 Multicast DNS Advertisement~~

~~During the establishment of a session with the Miracast Source, the Miracast Sink publishes a Multicast DNS (mDNS) [RFC6762] advertisement in the following format:~~

~~<instance name>.<service name>.<transport protocol>.<domain name>~~

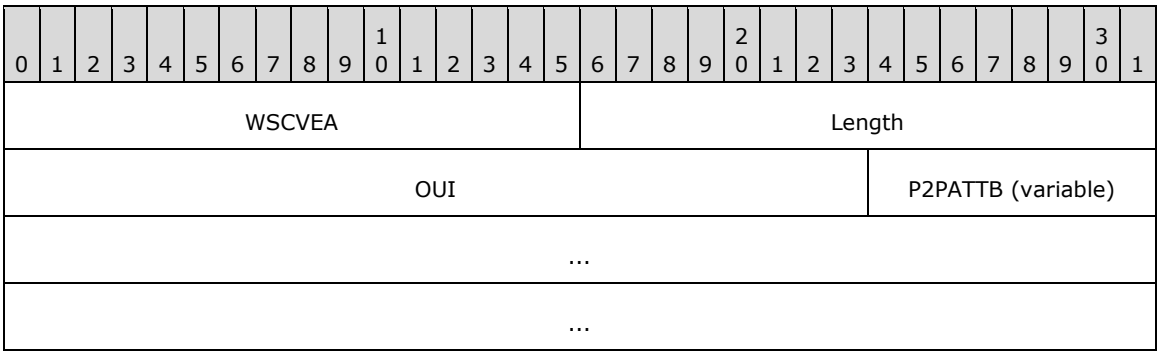
~~Where: <instance name> is the friendly name of the Sink; <service name> is "_display"; <transport protocol> is "_tcp"; and <domain name> is "local".~~

### ~~2.2.6~~2.2.4 Vendor Extension Attribute

The ~~Miracast over Infrastructure vendor extension~~**Vendor Extension** **a**ttribute is ~~published in the Wi-Fi Simple Configuration (~~a WSC~~)~~ information element (IE) ~~[WF-WSC2.0.2] for the Group Owner (GO) Beacons, Probe Request, and Probe Response frames ([WF-P2P1.2] section 4.2).~~

~~The~~structure that is used by a Miracast Sink to publish peer to peer (P2P) attribute ~~data is represented by one or more of the~~ structures defined ~~in the sections that follow~~by the Miracast over Infrastructure protocol.

~~The WSC vendor extension~~As specified in [WF-WSC2.0.2], the Vendor Extension attribute has the following general format~~:~~.




**WSCVEA (2 bytes):** The value is 0x1049 to indicate that this attribute is a WSC vendor extension.

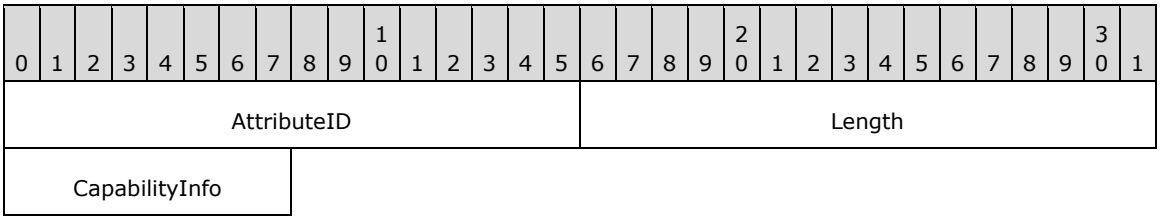
**Length (2 bytes):** The length of the following fields in bytes.

**OUI (3 bytes):** A Wi-Fi Protected Setup (WPS) organizationally unique identifier (OUI) [IEEE-OUI]. The value is 0x000137 for messages defined by this specification.

**P2PATTB (variable):** One or more of the ~~peer-to-peer (~~P2P~~)~~ attribute structures defined in the sections that follow. ~~The attributes~~Attributes can be included in any order.

### ~~2.2.6.1~~2.2.4.1 Capability Attribute

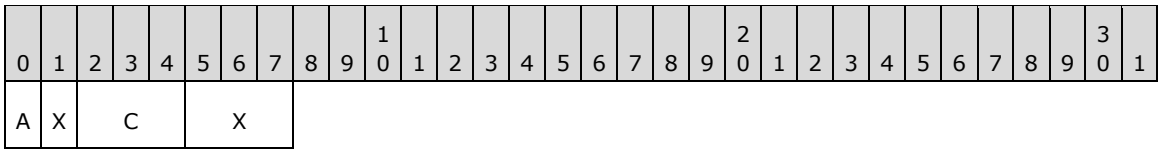
The **Capability** attribute indicates whether a connection over Miracast over Infrastructure is possible. This attribute MUST be present in the ~~vendor extension~~Vendor Extension attribute.




**AttributeID (2 bytes):** The Capability attribute ID, which is 0x2001.

**Length (2 bytes):** The length of the **CapabilityInfo** field, in bytes, which is 0x0001.

**CapabilityInfo (1 byte):** A bit field table with capability information, which has the following structure:




**A - MiracastOverInfrastructureSupport (1 bit):** 0 = not supported, 1 = supported.

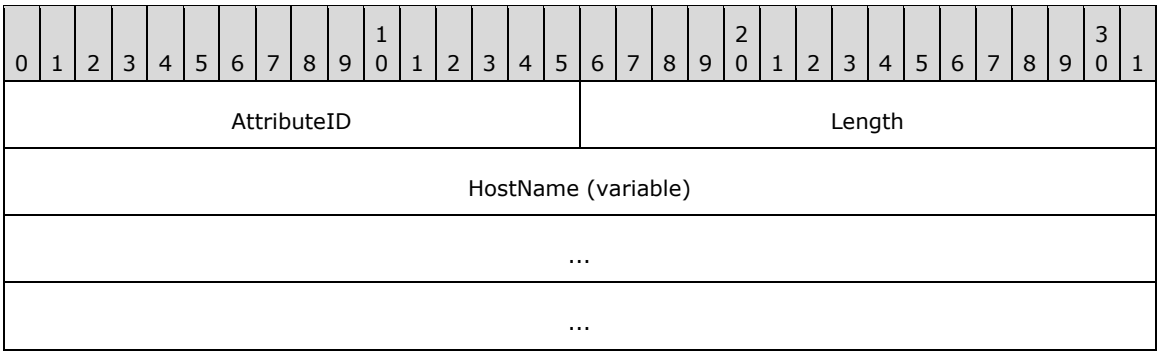
**X - Reserved (1 bit):** Reserved.

**C - Version (3 bits):** The version of this protocol, which is 0x1.

**X - Reserved (3 bits):** Reserved.

### ~~2.2.6.2~~2.2.4.2 Host Name Attribute

The **Host Name** attribute specifies the Miracast Sink host name. This attribute MUST be present exactly once in the ~~vendor extension~~Vendor Extension attribute.



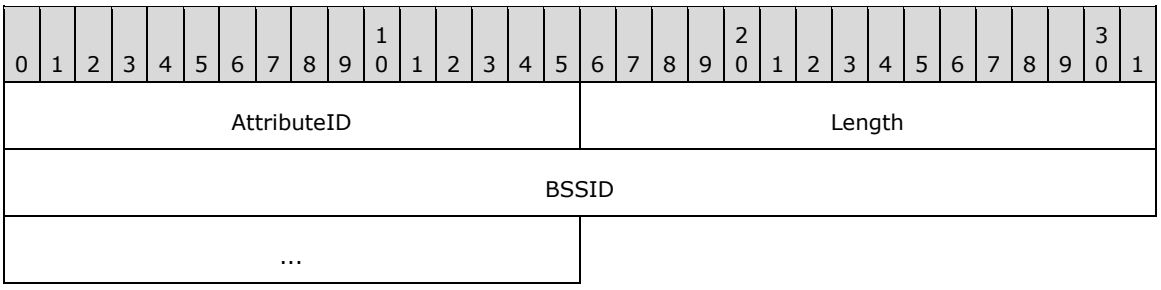

**AttributeID (2 bytes):** The Host Name attribute ID, which is 0x2002.

**Length (2 bytes):** The length of the **HostName** field, in bytes.

**HostName (variable):** The Miracast Sink host name string, encoded in ~~UTF-8~~ASCII. The host name is not fully qualified. A Sink having a host name that contains the ~~dot~~period ('.') character MUST NOT be used for Miracast over Infrastructure connections.

### ~~2.2.6.3~~2.2.4.3 BSSID Attribute

The **BSSID** attribute specifies the basic service set identifier (BSSID) for the 802.11 Access Point (AP) [IEEE802.11-2012] associated with the wireless network. This attribute is optional in the ~~vendor extension~~Vendor Extension attribute, but it MUST NOT appear more than once.



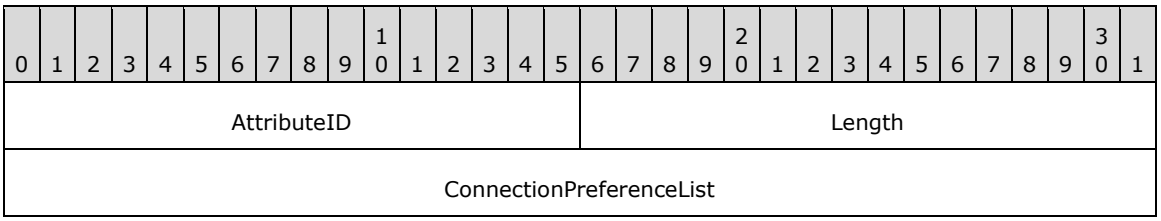

**AttributeID (2 bytes):** The BSSID attribute ID, which is 0x2003.

**Length (2 bytes):** The length of the **BSSID** field, in bytes, which is 0x0006.

**BSSID (6 bytes):** The BSSID for the associated WAP.

### ~~2.2.6.4~~2.2.4.4 Connection Preference Attribute

The **Connection Preference** attribute indicates the preference of transports for the connection of the Miracast Sink to the Miracast Source. ~~**This**~~The Sink MAY include a Connection Preference attribute ~~**is optional**~~ in the ~~**vendor extension**~~Vendor Extension attribute, but it MUST NOT appear more than once.



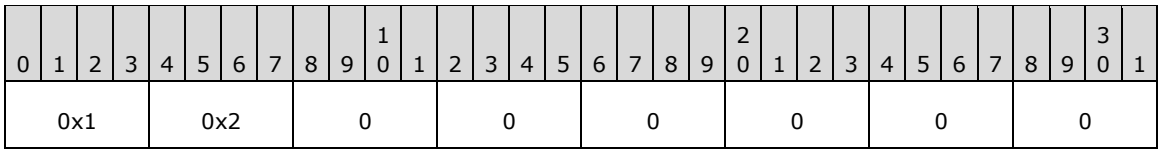

**AttributeID (2 bytes):** The Connection Preference attribute ID, which is 0x2004.

**Length (2 bytes):** The length of the **ConnectionPreferenceList** field, in bytes, which is 0x0004.

**ConnectionPreferenceList (4 bytes):** A packed array with room for 8 connection transport IDs, in descending order of preference. The following IDs are defined~~:~~.

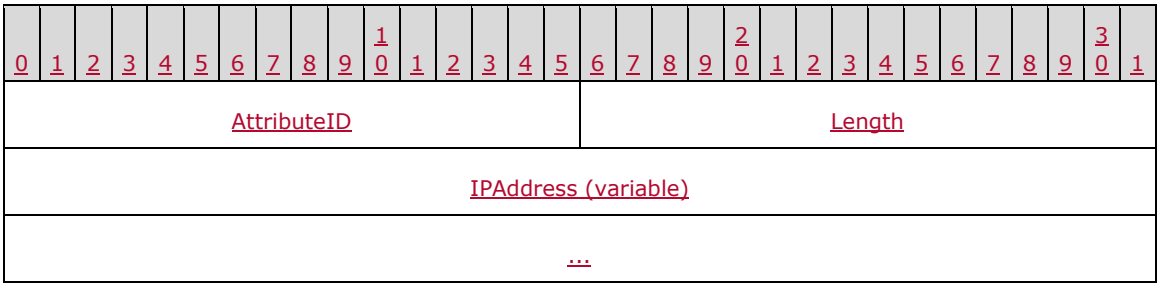
| Transport ID | Transport |
|---|---|
| 0x1 | Miracast over Infrastructure |
| 0x2 | Wi-Fi Direct (WFD) |

The following is an example of a preference list buffer with Miracast over Infrastructure preferred over WFD.




### 2.2.4.5 IP Address Attribute

The **IP Address** attribute specifies an IP address of the Miracast Sink. This attribute can occur zero or more times in the Vendor Extension attribute. The set of IP addresses included in the Vendor Extension attribute SHOULD<1> be the same set as the Sink's mDNS responder would provide to an mDNS requester.



...

**AttributeID (2 bytes):** The IP Address attribute ID, which is 0x2005.

**Length (2 bytes):** The length of the **IPAddress** field, in bytes.

**IPAddress (variable):** An IP address string, encoded in ASCII. The supported address formats are IPv4 in dotted decimal notation ([RFC1123] section 2.1) and IPv6 ([RFC4291] section 2.2).

# 3 Protocol Details

## ~~4~~3 ~~Miracast Sink Details~~

A Miracast over Infrastructure session consists of three phases: device discovery, host name resolution, and projection, as shown in the following diagram.

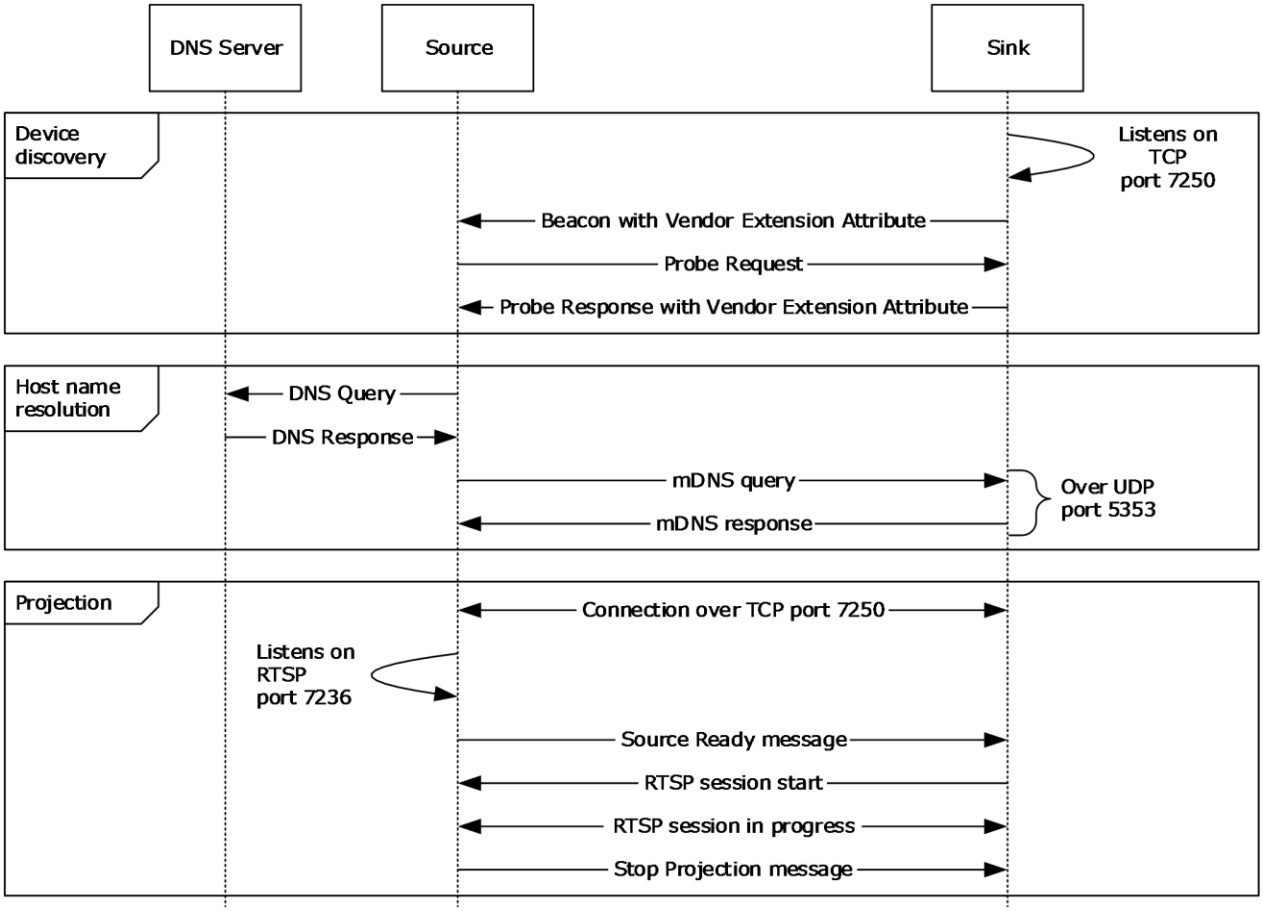



**Figure 2~~The~~: A Miracast over Infrastructure ~~attributes are published as a vendor extension attribute in the Wi-Fi Simple Configuration (~~session**

### Device discovery

The Miracast over Infrastructure session starts with peer to peer (P2P) device discovery ([WF-P2P1.2] section 3.1.2), which a Miracast Source uses to find a device capable of performing the functions of a Miracast Sink. This includes the Source sending Probe Request frames ([WF-P2P1.2] section 4.2.2) and listening for Probe Response frames ([WF-P2P1.2] section 4.2.3) and Beacon frames ([WF-P2P1.2] section 4.2.1).

Beacon frames are unsolicited broadcasts that advertise P2P devices. Probe Response frames are sent by a Sink in response to Probe Request frames sent by the Source. If the Source receives a Beacon or Probe Response that contains a WSC ~~) information element (~~IE~~)~~ [WF-WSC2.0.2] ~~from the Miracast Sink.~~ Vendor Extension attribute (section 2.2.4), the Source checks the Capability attribute (section 2.2.4.1) for Miracast over Infrastructure support.

If the Capability attribute specifies that Miracast over Infrastructure is not supported, the Source falls back to standard Miracast [WF-WSC2.0.2].

If one or more IP Address attributes (section 2.2.4.5) are included, the Source can skip host name resolution.

**Host name resolution**

The host name received by the Source during device discovery specifies the unqualified host name of the target Sink. The Source tries to resolve this host name by using DNS [IANA-DNS] [RFC1034] [RFC2181] and/or mDNS [RFC6762].

When host name resolution is complete, the session proceeds to the Projection phase.

The Source uses a **Discovery timer** (section 3.2.2) to limit the time it spends on host name resolution. If this timer reaches its timeout, the host name resolution fails, and the Source falls back to standard Miracast.

**Projection**

When the Source finds a device that can perform as the Sink, the Source attempts a connection to it over TCP port 7250, which it will use for sending Miracast over Infrastructure messages (section 2.2) to the Sink. These messages include starting and stopping the projection.

If the connection fails, the Source falls back to standard Miracast.

The Sink is expected to be listening for a Source Ready message (section 2.2.1) on TCP port 7250. When the Source is ready to project, it listens on Real-Time Streaming Protocol (RTSP) control port 7236 for a connection request, then it sends the Source Ready message. In turn, the Sink connects to the specified RTSP Source port to establish the link.

To pause or stop the projection, the Source sends a Stop Projection message (section 2.2.2) to notify the Sink. Upon receipt of that message, the Sink stops displaying the stream, and a disconnection follows from the Source on the socket that is connected on port 7250. The Source resumes projection by sending another Source Ready message to the Sink.

## 3.1 Miracast Sink Details

### ~~4.1.1~~3.1.1 Abstract Data Model

None.

### ~~4.1.2~~3.1.2 Timers

None.

### ~~4.1.3~~3.1.3 Initialization

Upon initialization, the Miracast Sink SHOULD register the following service instance name ([RFC6763] section 4.1) with the Sink's local mDNS implementation.

```
<instance name>. display. tcp.local
```

The <instance name> is the friendly name of the Sink, which will be associated with both port 7250 and the following TXT key-value pair ([RFC6763] section 6).

Key: container_id

Value: A GUID that identifies the Sink.

This service instance name is also used in [WF-DTS2.1] section 4.4.1.

After registering the service instance name, the Sink MUST start listening on TCP port 7250 for an inbound ~~session~~connection.

Finally, the Sink MUST begin being discoverable by Beacons and/or Probe Requests as in standard Miracast [WF-WSC2.0.2], except that every Beacon and Probe Response the Sink sends MUST include a Vendor Extension attribute (section 2.2.4).

### ~~4.1.4~~3.1.4 Higher-Layer Triggered Events

None.

### ~~4.1.5~~3.1.5 Message Processing Events and Sequencing Rules

~~There are two messages the Miracast Sink can receive, the Source Ready and Stop Projection messages (section 2.2.1).~~

#### 3.1.5.1 Receive Probe Request

When a Miracast Sink receives a Probe Request message, the Sink MUST send a Probe Response message [WF-P2P1.2] and include a WSC IE [WF-WSC2.0.2] Vendor Extension attribute (section 2.2.4) with a Capability Attribute (section 2.2.4.1) that indicates support for Miracast over Infrastructure.

#### 3.1.5.2 Receive Connection Request

When a Miracast Sink receives a new TCP connection attempt on port 7250, but it already has a TCP connection established, the Sink SHOULD reject the new connection request, but it MAY close the existing TCP connection instead and accept the new one.<2>

#### ~~4.1.5.1~~3.1.5.3 Receive Source Ready Message

When a Miracast Sink receives a Source Ready message (section 2.2.1~~.1~~), it MUST connect back to the ~~Miracast~~ Source over TCP on the RTSP port specified in the ~~Source Ready~~ message.

#### ~~4.1.5.2~~3.1.5.4 Receive Stop Projection Message

When ~~the~~a Miracast Sink receives a Stop Projection message (section 2.2.~~1.~~2), ~~the Sink SHOULD~~it MUST stop displaying the stream~~ and SHOULD expect a disconnection from the Miracast Source on the socket connected on port 7250~~.

### ~~4.1.6~~3.1.6 Timer Events

None.

### ~~4.1.7~~3.1.7 Other Local Events

~~A Miracast Sink might not receive~~If the RTSP connection receives a ~~Stop Projection~~teardown message ~~(section 2.2.1.2) at the end of a session. Therefore, a Sink MAY use other means~~as it does in standard Miracast, or if the connection to ~~determine that a Miracast~~the Source is ~~no longer projecting and clean up~~lost, or if the TCP port 7250 connection is lost, the Sink MUST close its session.

## ~~4.2~~3.2 Miracast Source Details

### ~~4.2.1~~3.2.1 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, provided their external behavior is consistent with that described in this document.

~~The~~A Source ID ~~value~~TLV (section 2.2.~~1.~~3.3) is maintained throughout the lifetime of the Miracast session. It is included in ~~each~~all Miracast ~~message~~messages (section 2.2~~.1~~) to identify the Miracast Source.

### ~~4.2.2~~3.2.2 Timers

This Miracast Source uses the following timers~~:~~.

**Discovery timer:** This timer is ~~initialized when~~used to limit the time the Source ~~attempts to resolve the~~spends on host name ~~of the~~resolution before giving up and falling back to standard Miracast ~~Sink over DNS or mDNS (section 3.2.5).<1>~~.

**Control channel connection timer:** This timer is ~~initialized when~~used to limit the time the Source ~~attempts~~spends waiting for the Sink to connect to ~~the~~it before giving up and falling back to standard Miracast ~~Sink over the control channel.<2>~~.

### ~~4.2.3~~3.2.3 Initialization

The Source ID ~~member~~ TLV (section 2.2.3.3) of the abstract data model (section 3.2.1) is initialized to an implementation-dependent value.

### ~~4.2.4~~3.2.4 Higher-Layer Triggered Events

~~None.~~

When a higher-layer application requests discovery of Miracast Sinks, the Miracast Source MUST send a standard Probe Request to Miracast Sinks within range, as specified in [WF-P2P1.2].

When the higher-layer application or protocol requests to disconnect the Miracast connection, the Source MUST send the Stop Projection message (section 2.2.2) to the Sink to stop the projection of the multimedia data stream. After sending this message, the Source MUST close the TCP session.

### ~~4.2.5~~3.2.5 Message Processing Events and Sequencing Rules

#### 3.2.5.1 ~~The~~Receive Beacon with Vendor Extension Attribute

When a Miracast Source ~~discovers~~receives a Beacon message that includes a Vendor Extension Attribute (section 2.2.4), it MUST check the ~~Miracast Sink over Wi-Fi Direct (WFD). WFD discovery is specified in~~ **MiracastOverInfrastructureSupport** bit in the ~~Wi-Fi Display Protocol [WF-DTS1.1] section 4.~~

~~During WFD discovery, the Source looks for the presence of the Wi-Fi Simple Configuration (WSC) information element (IE) vendor extension~~Capability attribute (section 2.2.~~3) and parses its contents. With information from this attribute, the Source initiates DNS and/or mDNS queries to resolve the host name of the target Sink device~~4.1), which indicates whether the Sink supports Miracast over Infrastructure.

If Miracast over Infrastructure is supported by the Sink, the Source MUST do the following.

1. If one or more IP Address attributes (section 2.2.4.5) are present in the message, the Source SHOULD<3> skip name resolution and treat the addresses as the result of host name resolution, by proceeding as specified in section 3.2.5.3; however, the Source MAY instead ignore them and continue as if they were not present.

2. If host name resolution was not skipped, the Source MUST do the following.

    1. Start its **Discovery timer** (section 3.2.2) to expire after an implementation-specific<4> period of time if host name resolution does not complete.

    2. Begin host name resolution on the name in the Host Name Attribute (section 2.2.4.2), using DNS and/or mDNS, the choice of which is implementation-specific.<5>

### 3.2.5.2 Receive Probe Response with Vendor Extension Attribute

When a Miracast Source receives a Probe Response message that includes a Vendor Extension Attribute (section 2.2.4), the Source MUST check the **MiracastOverInfrastructureSupport** bit in the Capability attribute (section 2.2.4.1), which indicates whether a Sink supports Miracast over Infrastructure.

If Miracast over Infrastructure is supported by the Sink, the Source MUST perform the actions specified in section 3.2.5.1, as if it had received a Beacon message indicating that the Sink supports Miracast over Infrastructure.

### 3.2.5.3 Host Name Resolution Complete

When a Miracast Source obtains a set of one or more IP addresses of the Miracast Sink, the Source MUST do the following.

1. Cancel its **Discovery timer** (section 3.2.2).

2. Start its **Control Channel Connection timer**, which will expire after an implementation-specific<6> time unless it receives a connection over the RTSP control channel.

3. Attempt a connection to one of the IP addresses over TCP port 7250. The method of choosing a Sink IP address is implementation-specific.<7>

### 3.2.5.4 Miracast Connection Complete

When the connection to the Sink over TCP port 7250 fails, the Source MUST do the following:

1. Abandon its attempt to start a Miracast over Infrastructure session.

2. Fall back to using standard Miracast [WF-WSC2.0.2].

If the connection attempt succeeds, the Source MUST do the following.

1. Begin listening on RTSP control port 7236 for a connection request.

### ~~4.2.5.1~~ Send ~~Source Ready Message~~

~~1.~~2. ~~When the Miracast Source has resolved the host name for the target Miracast Sink, the Source sends the Source~~ a Source Ready message (section 2.2.1~~.1) to the Sink. This message includes a specific RTSP port for the Sink to connect back on~~) over the TCP session.

~~4.2.5.2 Send Stop Projection Message~~

### 3.2.5.5 RTSP Connection Accepted

When a Miracast Sink accepts an RTSP connection, the Miracast Source ~~ends a session,~~MUST do the ~~Source sends a Stop Projection message~~following.

- Cancel the **Control Channel Connection** timer (section 3.2.2~~.1.2) to the Miracast Sink. After the message is sent, the Source closes the TCP session to the RTSP port~~).
- Perform standard RTSP behavior.

## ~~4.2.6~~3.2.6 Timer Events

If either of the Miracast Source timers (section 3.2.2) reaches its timeout, the ~~attempt to start a Miracast over Infrastructure session is abandoned and an WFD connection is attempted instead. This is shown in a figure in section 1.3~~Source MUST do the following.

1. Abandon its attempt to start a Miracast over Infrastructure session.
2. Fall back to using standard Miracast [WF-WSC2.0.2].

## ~~4.2.7~~3.2.7 Other Local Events

None.

# ~~5~~4 Protocol Examples

~~WSC~~ The following sections describe examples of Miracast over Infrastructure structures, which were taken from network captures of the protocol.

## ~~5.1~~4.1 Vendor Extension Attribute Example

~~10 49 // WSC~~This is an example of the Vendor Extension attribute (section 2.2.4).

```
10 49       // WSC Vendor Extension attribute
00 19 // Size =      // Length (25 Bytesbytes)
00 01 37 // Microsoft    // OUI (WPS OUIID)

20 01       // Capability Attributeattribute
00 01 // Size =      // Length (1 Bytebyte)
88          // Capability info

20 02 // HostName Attribute      // Host Name attribute
00 0F // Size =      // Length (15 Bytesbytes)
44 75 6D 6D 79 31 2D 4B 61 62 79 6C 61 6B 65 // HostName = "Dummy1-Kabylake"
```

## ~~5.2~~ Source Ready Message Example

```
00 3D  // Size = 61 bytes
01  // Version = 0x1
```

## ~~4.2 01 //~~

This is an example of the Source Ready message (section 2.2.1).

```
00 3D       // Length (61 bytes)
01          // Version
01          // SOURCE_READY

00          // Friendly Name TLV
00 1E  // Size =      // Length (30 bytes)
44 00 75 00 6D 00 6D 00 79 00 31 00 2D 00 4B 00  //  // "Dummy1-Kabylake"
61 00 62 00 79 00 6C 00 61 00 6B 00 65 00

02          // RTSP Port TLV
00 02 // Size =      // Length (2 Bytesbytes)
1C 44       // Port = (7236)

03          // Source IdID TLV
00 10 // Size =      // Length (16 Bytesbytes)
91 F4 AB E9 EF F5 46 4A AE E2 69 72 2A ED 11 B5 // Source ID
```

## ~~5.3~~4.3 Stop Projection Message Example

This is an example of the Stop Projection message (section 2.2.2).

```
00 38       // Size = (56 bytes)
01          // Version = 0x1
02 // Stop Projection

02          // STOP_PROJECTION
```

```
00          // Friendly Name TLV
00 1E ~~// Size =~~ // Length (30 bytes)
44 00 75 00 6D 00 6D 00 79 00 31 00 2D 00 4B 00 ~~//~~ // "Dummy1-Kabylake"
61 00 62 00 79 00 6C 00 61 00 6B 00 65 00

03          // Source ~~Id~~ID TLV
00 10 ~~// Size =~~ // Length (16 ~~Bytes~~bytes)
91 F4 AB E9 EF F5 46 4A AE E2 69 72 2A ED 11 B5 // Source ID
```

## ~~6~~5 Security Considerations

~~It is recognized that~~Because a Miracast over Infrastructure ~~stream can occur without the benefit~~session has no security of its own, use it only when security is provided at another layer, such as link layer ~~encryption when the connection is not over Wi-Fi Direct (WFD). <3~~security (WPA2) on a wireless network, or physical security on a wired network.<8>

# ~~7~~6 Appendix A: 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~~updates to those products.

- Windows 10 v1703 operating system

Exceptions, if any, are noted ~~below.~~in this section. If ~~a~~ an update version, service pack or ~~Quick Fix Engineering (QFE~~Knowledge Base (KB) number appears with ~~the~~a product ~~version,~~name, the behavior changed in that ~~service pack or QFE.~~update. The new behavior also applies to subsequent ~~service packs of the product~~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 ~~3.~~2.2.4.5: The Windows Sink implementation adds a single IP Address attribute with an IPv4 address.

<2> Section 3.1.5.2: The Windows Sink implementation rejects new connections except on the Surface Hub SKU, where the new connection replaces the existing one.

<3> Section 3.2.5.1: The IP Address attribute (section 2.2.4.5) is not supported in Windows 10 v1709 operating system and earlier implementations.

<4> Section 3.2.5.1: The Windows implementation uses a period of 1.5 second for the ~~discovery~~**Discovery** **timer**.

~~<2> Section 3.2.2: The Windows~~ <5> Section 3.2.5.1: The Windows implementation attempts both DNS and mDNS in parallel and uses the first one to respond; however, when connected to a virtual private network (VPN), mDNS is preferred, so if the Windows implementation gets DNS results, it still waits for mDNS to complete or time out.

<6> Section 3.2.5.3: The Windows implementation uses a period of 5 seconds for the ~~control channel connection~~ **Control Channel Connection** **timer**.

~~<3> Section 5:~~<7> Section 3.2.5.3: The Windows implementation of the Source chooses the first IP address in the set.

<8> Section 5: The Windows implementation does not attempt a Miracast over Infrastructure connection over a wireless network, if the wireless network it is connected to does not employ link layer security (WPA2).

# ~~8~~7 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 |
|---|---|---|
| 1.3 Overview | 8371 : Added support for skipping source name resolution. | Major |
| 2.2.3.1 Friendly Name TLV | 8371 : Defined the maximum length of the friendly name string. | Major |
| 2.2.4.5 IP Address Attribute | 8371 : Added the IP Address vendor extension attribute. | Major |
| 3 Protocol Details | 8371 : Added a detailed diagram and summary of the behavior of a Miracast over Infrastructure session. | Major |
| 3.1.5.1 Receive Probe Request | 8371 : Added Receive Probe Request section. | Major |
| 3.1.5.2 Receive Connection Request | 8371 : Added Receive Connection Request section. | Major |
| 3.2.5 Message Processing Events and Sequencing Rules | 8317 : Added support for skipping source name resolution. | Major |
| 3.2.5.1 Receive Beacon with Vendor Extension Attribute | 8371 : Added Receive Beacon with Vendor Extension Attribute section. | Major |
| 3.2.5.2 Receive Probe Response with Vendor Extension Attribute | 8371 : Added Receive Probe Response with Vendor Extension Attribute section. | Major |
| 3.2.5.3 Host Name Resolution Complete | 8371 : Added Host Name Resolution Complete section. | Major |
| 3.2.5.4 Miracast Connection Complete | 8371 : Added Miracast Connection Complete section. | Major |
| 3.2.5.5 RTSP Connection Accepted | 8371 : Added RTSP Connection Accepted section. | Major |

# ~~98~~ Index

**N**



**O**



**P**



**R**

## S



## T



## V