Microsoft Windows

A WINDOWS® RALLY™ SPECIFICATION

Windows Connect Now–NET

Abstract

Microsoft® Windows® Connect Now technology enables simple and secure configuration of wireless networks and provisioning of wireless hardware. Windows Connect Now-NET (WCN-NET) is the Microsoft implementation of the Simple Configuration Protocol, a new standard in the Wi-Fi Alliance. WCN-NET supports configuration of devices on out-of-band Ethernet and in-band wireless networks.

Windows Connect Now-NET in Microsoft Windows Vista™ communicates with access points and wireless stations by using Universal Plug and Play (UPnP), authenticates with them by using a personal identification number (PIN), and provides wireless settings that are based on user selection.

This specification defines the WCN-NET implementation details for devices that connect with systems running the Windows Vista operating system. WCN-NET is a component of the Microsoft Windows Rally™ set of technologies.

Version 1.1 December 8, 2006

Disclaimer

This is a preliminary document and may be changed substantially prior to final commercial release of the software described herein.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

The current version of this specification is maintained on the Web at:

http://www.microsoft.com/rally

Revision History

Date	Revision
May 8, 2006	Version 1.0
December 8, 2006	Version 1.1

Contents

Introduction to WCN-NET

Microsoft® Windows® Connect Now technology provides solutions for creating secure wireless networks and adding devices to the network. Specifically, Windows Connect Now-NET (WCN-NET) solves two problems that have limited consumer deployment of secure wireless networks:

- Most users do not realize that the default network configuration is not secure.
- Many of the remaining users find that the security configuration is too complex.

WCN-NET solves these problems by providing a user-friendly, simplified, and consistent way to set up secure wireless networks and add devices to the network. This solution works for both out-of-band Ethernet devices and in-band wireless devices

This specification summarizes the architecture and then covers registration in detail:

- User interface flow
- Registration Protocol
- Message format
- Registration message attributes
- Transportation of Registration Protocol messages by using universal Plug and Play (UPnP) or Extensible Authentication Protocol (EAP)

Appendix A explains the master table definitions. References and resources discussed in this specification are listed in “Resources” at the end of this specification.

WCN-NET User Experience

Entry Points and Flows

By using WCN-NET, device configuration and setup can be done through three entry points:

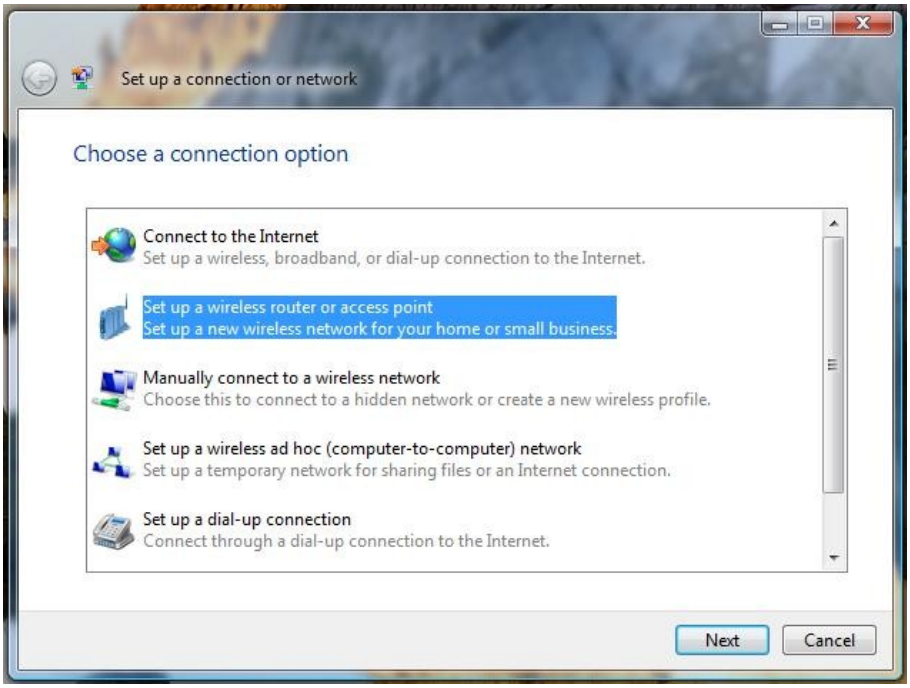
- Set up a wireless access point or use the Router Wizard.
- Use the Add a Wireless Device Wizard.
- Double-click the device in Network Explorer.

Set up a Wireless Access Point or Use the Router Wizard

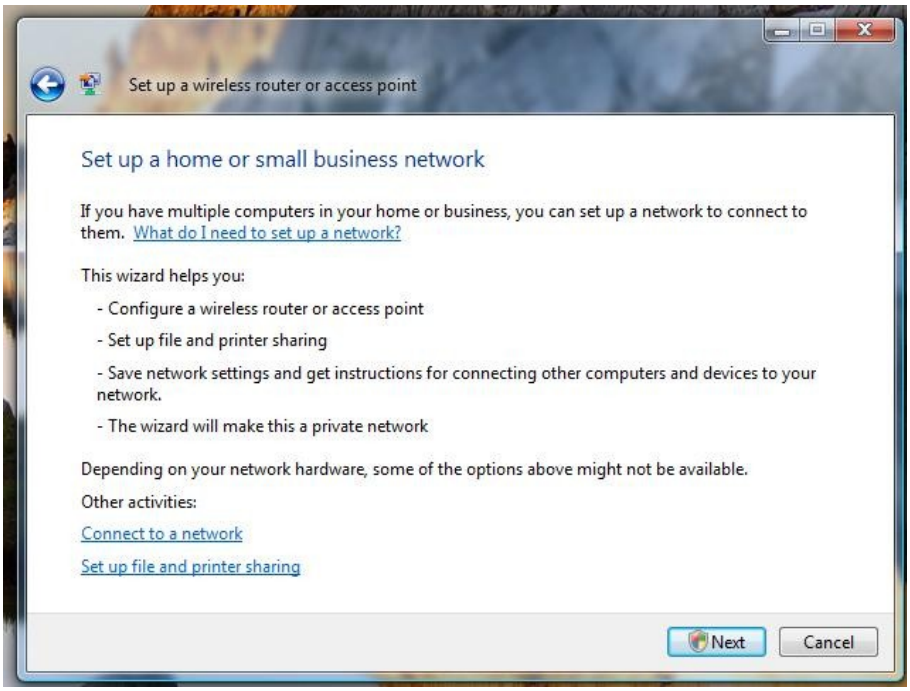
This wizard is targeted for first-time wireless access point and network setup. It helps users to set up most common network settings and to set up a wireless access point.

To run the wizard:

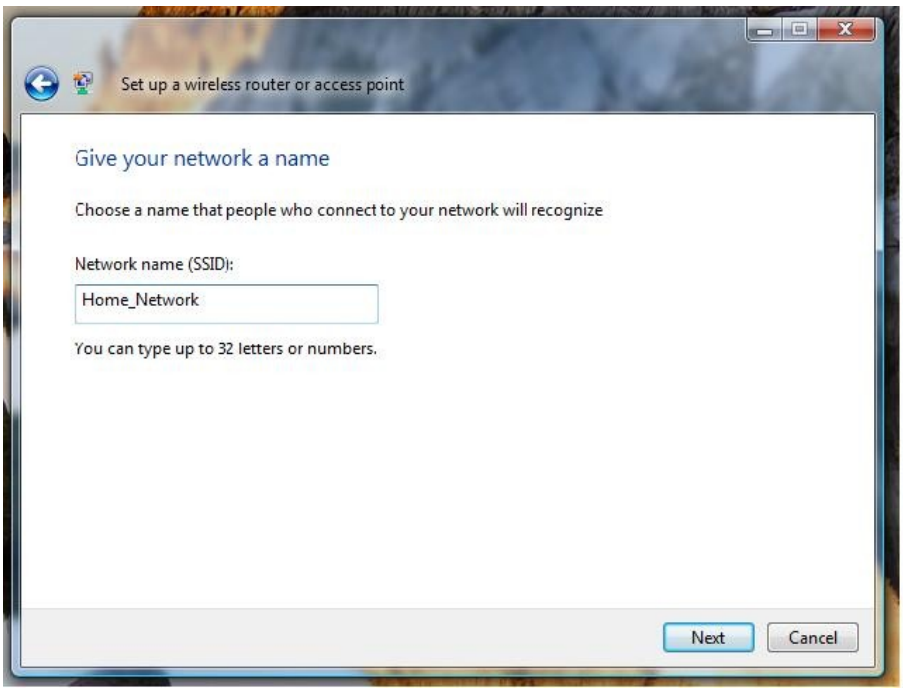
1. On the taskbar, click **Start**, click **Network**, click **Network and Sharing Center**, and then click **Set up a connection or network**.
   The **Choose a connection option** page appears.
2. Click **Set up a wireless router or access point**
   Set up a new wireless network for your home or small business.



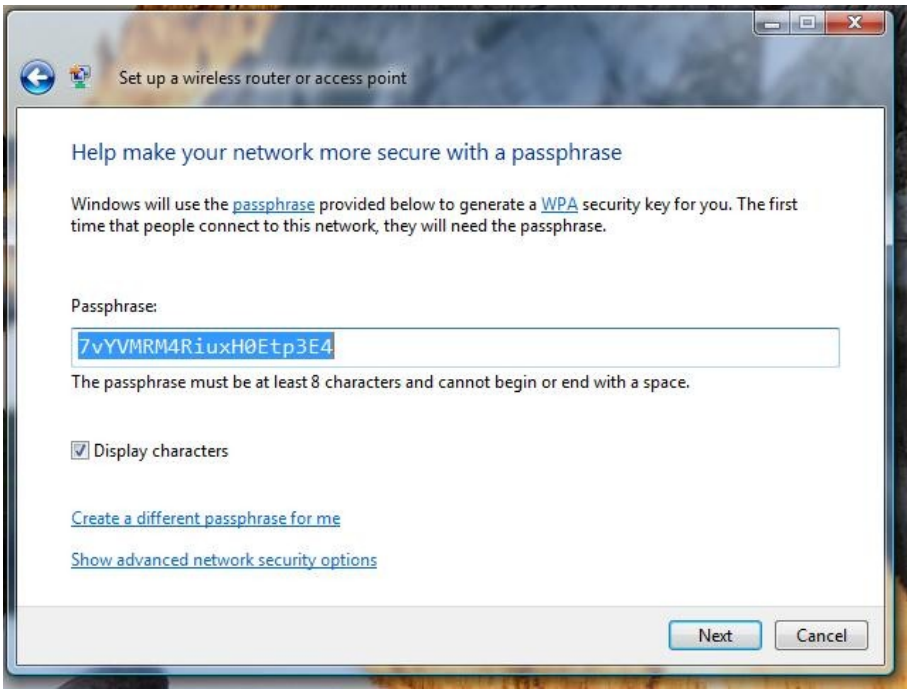
3. The introduction page appears, describing the detailed steps of the wizard. Click **Next**.



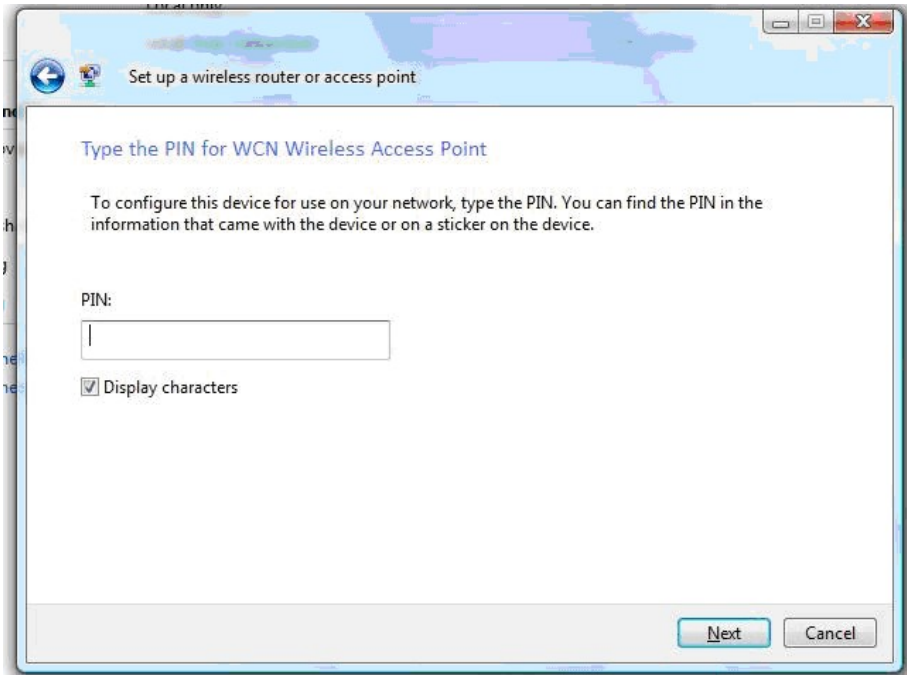
4. If the wizard detects a device, a **preselected Network Name (SSID)** appears. You can edit this field by typing a new name. Click **Next**.



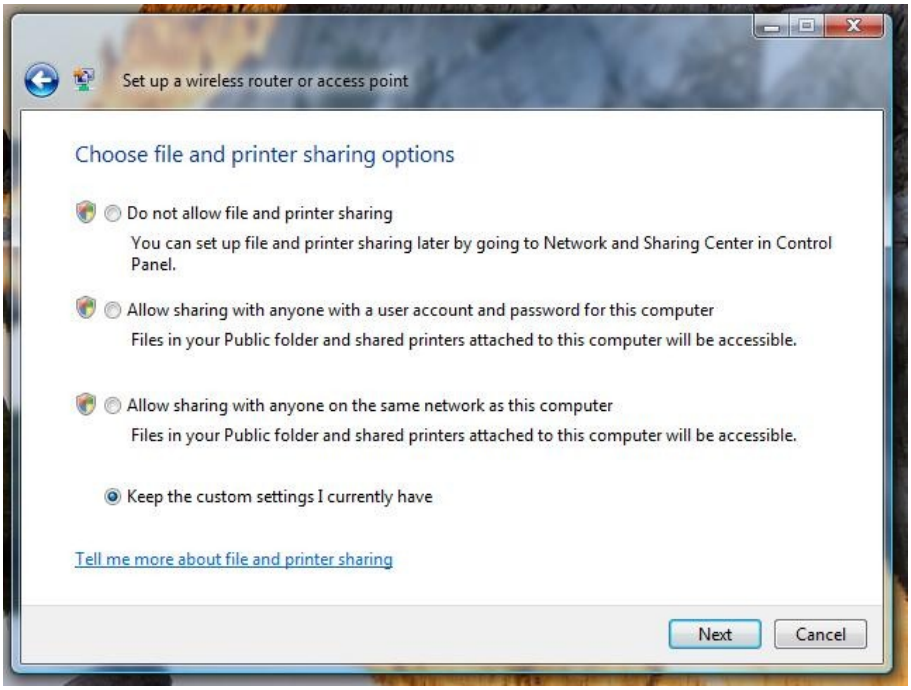
5. A preselected **Passphrase** appears. You can edit this field by typing a new name. Click **Next**.



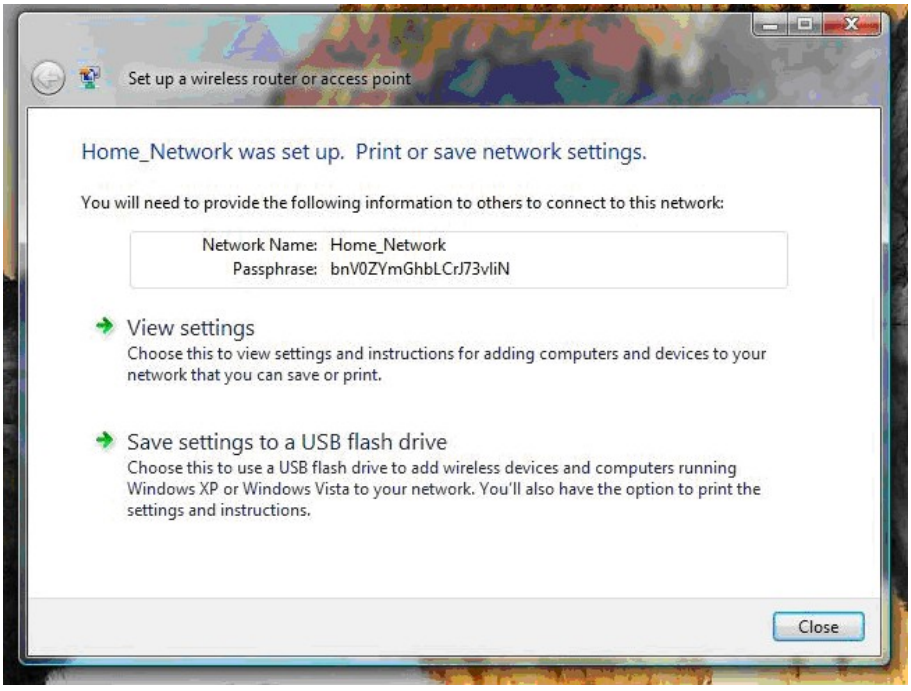
6. To continue the configuration process, type a device **PIN**. Click **Next**.



7. Configure commonly used file and printer sharing settings. Click **Next**.



8. Configuration is completed successfully. You can save and print these settings. Click **Close**.

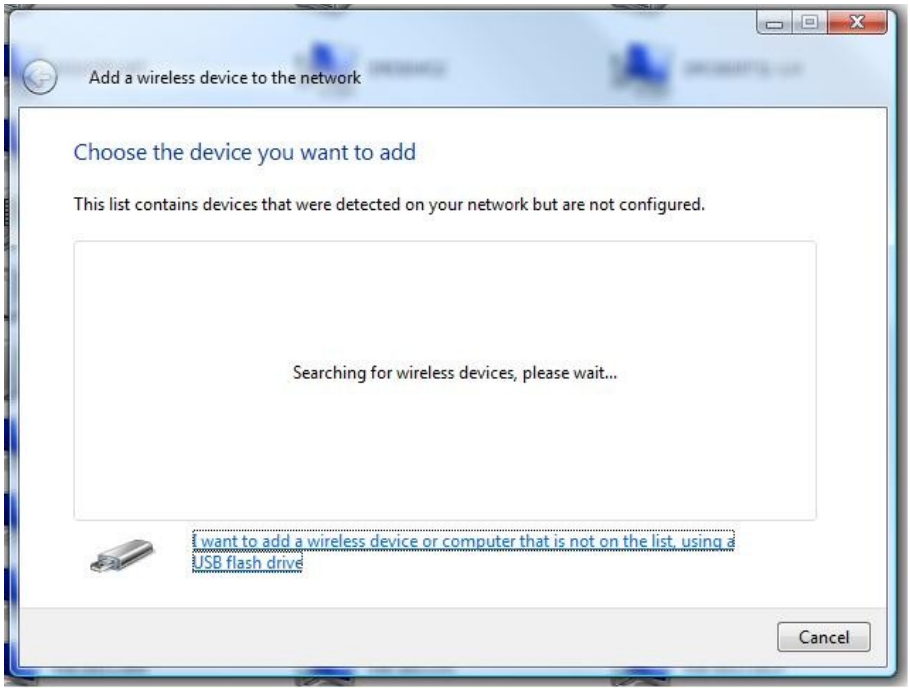


Use the Add a Wireless Device Wizard

This wizard is optimized for adding or setting up wireless devices for an existing network. However, users can also set up a new wireless network.

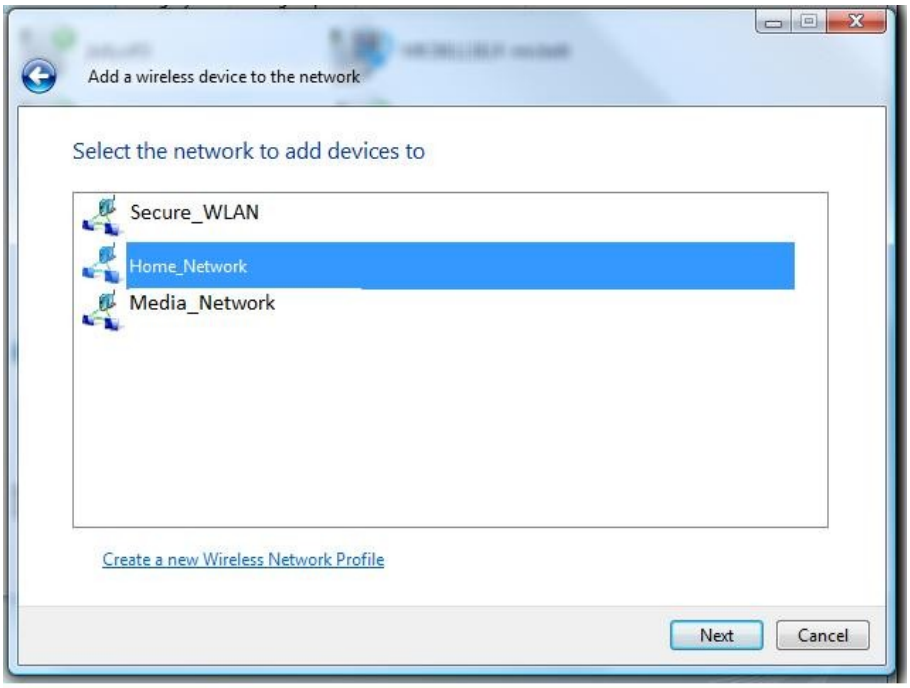
To run the wizard:

1. On the taskbar, click **Start**, click **Network**, and then click **Add a Wireless Device**.
   The discovered wireless devices that support WCN-NET appear in this device picker.



2. Choose the device that you want to add.

3. Complete the configuration process by creating a new wireless network and using the device PIN. You can also select existing profiles by using a profile picker.



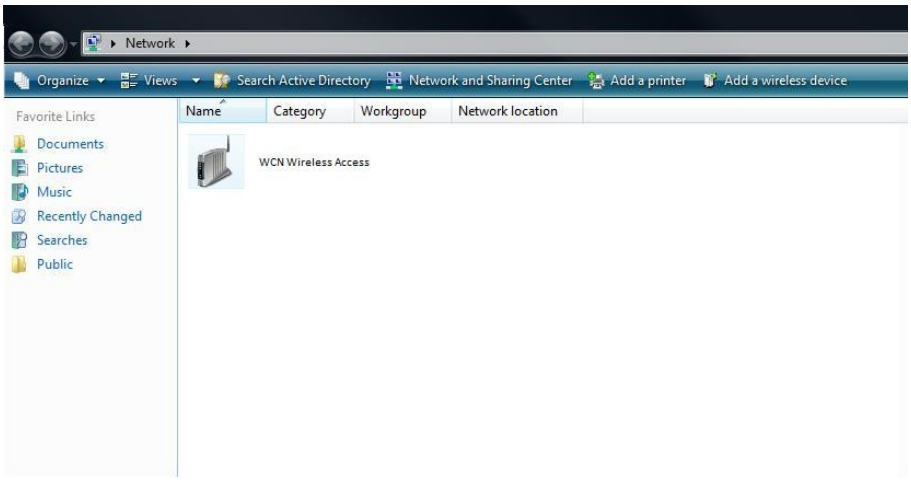
Double-Click the Device in Network Explorer

A Windows Connect Now device in Network Explorer has a default action of configure. Double-clicking the device allows the user to launch the configuration process and set up the device that supports WCN-NET.

To configure the device

1. On the taskbar, click **Start**, click **Network**, and then double-click **Access Point**.

2. Complete the device configuration process by creating a new wireless network, using the device PIN, or selecting or creating a wireless profile.



3. After the selection of the device, the configuration process can be completed by creating or selecting a wireless network.

User Experience Pamphlet in the Box for Access Points

For Access Points:

Instructions should be provided to the user for setting up a new wireless access point.

1. On the taskbar, click **Start**, click **Network**, and then click **Network and Sharing Center**.
2. Click **Set up a connection or network**, click **Set up a wireless router or access point**, and then click **Next** to complete the configuration.

For other wireless devices:

Instructions should be provided to the user for setting up a wireless device.

1. On the taskbar, click **Start**, click **Network**, click **Network and Sharing Center**, and then click **Add a Wireless**.
2. Select your wireless device, and then click **Next** to complete the configuration.

WCN-NET Architecture

Figure 1 shows the logical components of the WCN-NET architecture.

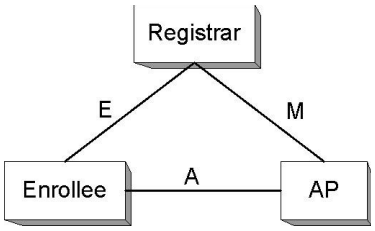


Figure 1. WCN-NET Components

The enrollee is a new device that does not have the settings for the wireless network. The registrar provides wireless settings to the enrollee. The access point provides normal wireless network hosting and also proxies messages between the enrollee and the registrar.

In Windows Vista, a new enrollee may exchange messages directly with the Windows Vista registrar (interface E) via UPnP if the enrollee is initially connected to an Ethernet network. Alternatively, a new enrollee may exchange messages over EAP with the Windows Vista Registrar and the access point works as a proxy to convey the messages to UPnP.

The message exchange between the registrar and the enrollee to authenticate and provide the enrollee with network settings is called the Registration Protocol.

Registration in Windows Vista

The registrar in Windows Vista is initiated via one of two methods:

- Opening Network Explorer.
- From the Network Center, clicking **Set up a connection or network** and then clicking **Set up a wireless router or access point**.

When the Windows Vista registrar process starts, it discovers all UPnP devices on the network and subscribes to UPnP events from any access points. It waits for UPnP events from access points and lists WCN-NET devices as it finds them.

In Windows Vista, Network Explorer presents a list of discovered devices, including WCN-NET-based devices that the user can select to configure. Clicking **Add a wireless device** in Network Explorer lists only unconfigured wireless devices.

Alternatively, if the user chooses to use the Network Center to create a new network by using **Set up a connection or network**:

- Windows displays a list of devices that are visible on the network. The user can select one of these devices to configure.
- Then the user is prompted to enter the device's PIN, which is used when authenticating between the Windows Vista registrar and the device.
- The user can then either select an existing network profile that contains a service set identifier (SSID) and passphrase or create new network settings if a profile does not already exist for the settings to be provided to the device.

- After the PIN and the network settings have been collected from the user, the Registration Protocol then runs between the Windows Vista registrar and the device.
- The PIN is used for two-way authentication, and the selected and defined profile is provided to the device.
- Upon successful completion of the Registration Protocol, the Windows Vista registrar displays a message to show that the device was successfully configured for the network.

The specifics of the WCN-NET protocol, including registration, are detailed in this specification.

Registration Protocol

The Registration Protocol provides:

- Two-way discovery
- Exchange of Diffie-Hellman public keys
- Lock-step message exchange
- Two-way authentication
- Transfer of configuration

Figure 2 describes the Registration Protocol message exchange.

Enrollee → Registrar:	M_1 =	Version || N1 || Description || PK_E
Enrollee ← Registrar:	M_2 =	Version || N1 || N2 || Description || PK_R [|| ConfigData] || $HMAC_{AuthKey}(M_1 \|\| M_2^*)$
Enrollee → Registrar:	M_3 =	Version || N2 || E-Hash1 || E-Hash2 || $HMAC_{AuthKey}(M_2 \|\| M_3^*)$
Enrollee ← Registrar:	M_4 =	Version || N1 || R-Hash1 || R-Hash2 || $ENC_{KeyWrapKey}$(R-S1) || $HMAC_{AuthKey}$ ($M_3 \|\| M_4^*$)
Enrollee → Registrar:	M_5 =	Version || N2 || $ENC_{KeyWrapKey}$(E-S1) || $HMAC_{AuthKey}$ ($M_4 \|\| M_5^*$)
Enrollee ← Registrar:	M_6 =	Version || N1 || $ENC_{KeyWrapKey}$(R-S2) || $HMAC_{AuthKey}$ ($M_5 \|\| M_6^*$)
Enrollee → Registrar:	M_7 =	Version || N2|| $ENC_{KeyWrapKey}$(E-S2 [||ConfigData]) || $HMAC_{AuthKey}$ ($M_6 \|\| M_7^*$)
Enrollee ← Registrar:	M_8 =	Version || N1 || [$ENC_{KeyWrapKey}$(ConfigData)] || $HMAC_{AuthKey}$ ($M_7 \|\| M_8^*$)

Figure 2: Registration Protocol Message Exchange

The following defines the conventions that were used in Figure 2:

||
Concatenation of parameters to form a message.

Subscripts
When used in the context of a cryptographic function such as HMACKey, a reference to the key that the function uses.

M_n^*
Message Mn excluding the HMAC-SHA-256 value.

Version
The type of Registration Protocol message.

N1
A 128-bit random number (nonce) that the enrollee specifies.

N2
A 128-bit random number (nonce) that the registrar specifies.

Description
A human-readable description of the sending device (UUID, manufacturer, model number, MAC address, and so on) and device capabilities such as supported algorithms, I/O channels, and Registration Protocol role. Description data is also included in 802.11 Probe request and Probe response messages.

PK_E and PK_R
Diffie-Hellman public keys of the enrollee and registrar, respectively.

AuthKey
An authentication key that is derived from the Diffie-Hellman secret gABmod p, the nonces N1 and N2, and the enrollee's MAC address.

E-Hash1 and E-Hash2
Precommitments that the enrollee makes to prove knowledge of the two halves of its own device password.

R-Hash1 and R-Hash2
Precommitments that the registrar makes to prove knowledge of the two halves of the enrollee's device password.

$ENC_{KeyWrapKey}$(...)
Symmetric encryption of the values in parentheses by using the key KeyWrapKey. The encryption algorithm is AES-CBC.

R-S1 and R-S2
Secret 128-bit nonces that, together with R-Hash1 and R-Hash2, the enrollee can use to confirm the registrar's knowledge of the first and second half, respectively, of the enrollee's device password.

E-S1, E-S2
Secret 128-bit nonces that, together with E-Hash1 and E-Hash2, can the registrar can use to confirm the enrollee's knowledge of the first and second half of the enrollee's device password, respectively.

$HMAC_{AuthKey}$(...)
An authenticator attribute that contains an HMAC keyed hash over the values in parentheses and using the key AuthKey. The keyed hash function is HMAC-SHA-256.

ConfigData
Wireless local area network (WLAN) settings and credentials. The registrar encrypts WLAN settings.

Summary and Classification of Keys

Table 1 lists the different keys that Windows Vista uses.

Table 1. Summary and Classification of Keys

Key name	Type	Known by	Used for
PK_E	Authentication and key derivation, long-lived or temporary	Enrollee and registrar	Generating session keys
PK_R	Authentication and key derivation, long-lived or temporary	Enrollee and registrar	Generating session keys
Device PIN	Authentication, temporary if shown on display, may be long-lived if on label	Enrollee and registrar	Authenticating Diffie-Hellman exchange
g^{AB}mod p	Authentication and key derivation, temporary	Enrollee and registrar	Generating session keys
KDK	Key derivation, temporary	Enrollee and registrar	Generating session keys
AuthKey	Authentication, temporary	Enrollee and registrar	Mutual authentication of enrollee and registrar
KeyWrapKey	Key wrap, temporary	Enrollee and registrar	Encrypting WLAN configuration for enrollee
PSK1	Authentication, temporary	Enrollee and registrar	Proof-of-possession of device password
PSK2	Authentication, temporary	Enrollee and registrar	Proof-of-possession of device password
EMSK	Key derivation, temporary	Enrollee and registrar	Not used

Key Derivation

Upon receipt of M1, the registrar has enough information to determine whether to use the in-band or out-of-band method for enrollment. The Registration Protocol message exchange applies the following rules for deriving security keys:

- If M2 is sent over a physically secure out-of-band channel, then **ConfigData** can be sent in M2 and the Registration Protocol can terminate at that point.
- Depending upon the physical security of the out-of-band channel and the registrar's policy, the registrar can choose whether to encrypt **ConfigData** that is sent in an out-of-band M2. Encrypting this data provides an additional measure of security.

1536-bit MODP Group for Diffie-Hellman Exchange

The 1536 bit MODP group that WCN-NET uses is taken from RFC 3526.

The prime is: 2^1536 - 2^1472 - 1 + 2^64 * { [2^1406 pi] + 741804 }

Its hexadecimal value is as follows:

```
FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1
29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED
EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D
C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F
83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D
670C354E 4ABC9804 F1746C08 CA237327 FFFFFFFF FFFFFFFF
```

The generator is: 2.

Derivation of KDK

$$KDK = HMAC\text{-}SHA\text{-}256_{DHKey}(N1 \,||\, EnrolleeMAC \,||\, N2)$$

DHKey is defined as SHA-256(g^{AB}mod p). PK_E is g^Amod p and PK_R is g^Bmod p. The enrollee and registrar know the secret values A and B, respectively. EnrolleeMAC is the 6-byte 802.11 MAC address of the enrollee. The enrollee's MAC address is included in the description data that is sent in M1.

Derivation of AuthKey, KeyWrapKey, and EMSK

Additional keys are derived from KDK by using a key derivation function (kdf). The function **prf** that is used in **kdf** is the keyed hash HMAC-SHA-256.

```
kdf(key, personalization_string, total_key_bits) :
    result := ""
    iterations = (total_key_bits + prf_digest_size – 1)/prf_digest_size
    for i = 1 to iterations do
    result := result || prf(key, i || personalization_string || total_key_bits)
    return 1st total_key_bits of result and destroy any bits left over
```

Given KDK and this key derivation function, the Registration Protocol session keys are derived as follows:

```
AuthKey || KeyWrapKey || EMSK = kdf(KDK, "Wi-Fi Easy and Secure Key
Derivation", 640)
```

AuthKey **(256 bits)**
: A key that is used to authenticate the Registration Protocol messages.

KeyWrapKey **(128 bits)**
: A key that is used to encrypt secret nonces and **ConfigData**.

EMSK **(256 bits)**
: An extended master session key that may be used to derive application-specific keys.

This notation means that 640 bits are generated by the kdf function by using the seed value KDK. These 640 bits are split into three parts that correspond to the two symmetric session keys AuthKey and KeyWrapKey and the EMSK.keying material.

Key Wrap Algorithm

The following algorithm is used to perform the key wrap function that is used to protect the secret nonces and the ConfigData:

1. First compute **KWA** = 1st 64 bits of $HMAC_{AuthKey}$(**DataToEncrypt**)
2. Generate random 128-bit **IV**.
3. Compute WrappedData = AES-Encrypt-$CBC_{KeyWrapKey}$(DataToEncrypt || KWA, IV).
4. **IV** is included along with **WrappedData** in the **Encrypted Settings** attribute.

To decrypt, use the following algorithm:

1. Data || KWA = AES-Decrypt-$CBC_{KeyWrapKey}$(WrappedData, IV)
2. If **KWA** = 1st 64 bits of $HMAC_{AuthKey}$(**Data**), then output **Data** or else output "failure".

Note that the IV must be random, and it must not be copied from any keying material that is used for other purposes. A freshly-generated random nonce must be used. KWA is the **Key Wrap Authenticator** attribute.

PIN Proof of Possession

E-Hash1 is derived from the session parameters and the device password. First, the device password is converted to two 128-bit PSK values as follows:

PSK1 = first 128 bits of $HMAC_{AuthKey}$(1st half of DevicePassword)

PSK2 = first 128 bits of $HMAC_{AuthKey}$(2nd half of DevicePassword)

The enrollee creates two 128-bit secret nonces (E-S1 and E-S2) and then computes:

E-Hash1 = $HMAC_{AuthKey}$(E-S1 || PSK1 || PK_E || PK_R)

E-Hash2 = $HMAC_{AuthKey}$(E-S2 || PSK2 || PK_E || PK_R)

The registrar creates two 128-bit secret nonces (R-S1 and R-S2) and then computes:

R-Hash1 = $HMAC_{AuthKey}$(R-S1 || PSK1 || PK_E || PK_R)

R-Hash2 = $HMAC_{AuthKey}$(R-S2 || PSK2 || PK_E || PK_R)

The hash values are gradually exchanged and verified in messages M3 through M7. If a verification check of one of the Device Password parts fails, the receiving side must acknowledge the message with a failure indication and the enrollee and registrar must stop the protocol and discard all keys and nonces that are associated with the session.

PIN Checksum

Windows Vista supports both 4- and 8-digit PINs. Only devices with displays can use the 4-digit PIN. Although the WCN-NET specification supports rekeying, Windows Vista does not.

The device password ID must be default, value = 0 (the device password is a PIN). For 8-digit numeric PINs, the last digit in the PIN is used as a checksum of the other digits. The algorithm to validate the checksum is given in the following C code.

```
bool ValidateChecksum(unsigned long int PIN)
{
  unsigned long int accum = 0;
        accum += 3 * ((PIN / 10000000) % 10);
        accum += 1 * ((PIN / 1000000) % 10);
        accum += 3 * ((PIN / 100000) % 10);
        accum += 1 * ((PIN / 10000) % 10);
        accum += 3 * ((PIN / 1000) % 10);
        accum += 1 * ((PIN / 100) % 10);
        accum += 3 * ((PIN / 10) % 10);
        accum += 1 * ((PIN / 1) % 10);

   return (0 == (accum % 10));
}
```

The corresponding algorithm to compute the checksum digit, assuming the other seven random PIN digits, is as follows:

```
int ComputeChecksum(unsigned long int PIN)
{
        unsigned long int accum = 0;
        accum += 1 * ((PIN / 1000000) % 10);
        accum += 3 * ((PIN / 100000) % 10);
        accum += 1 * ((PIN / 10000) % 10);
        accum += 3 * ((PIN / 1000) % 10);
        accum += 1 * ((PIN / 100) % 10);
        accum += 3 * ((PIN / 10) % 10);

        int digit = (accum % 10);
        return (10 - digit) % 10;
}
```

Message Format

WCN-NET specifies a message exchange protocol—Registration Protocol—between an enrollee and a registrar and several transports over which it can operate. The Registration Protocol is a concatenation of binary format attributes, in which each attribute uses a Type, Length, and Value structure as shown in Table 2.

Table 2. Type, Length, Value (TLV) Format for WCN-NET Binary Data

Byte offset	Field length (in bytes)	Field name	Description
0	2	AttributeType	Type identifier for the attribute
2	2	DataLength	Length in bytes of the attribute's data field
4	0-0xFFFF	Data	Attribute data

For a list of the attributes and associated definitions used in Windows Vista, see Appendix A.

Windows Vista provides a UPnP-based registrar function. It may be used for configuring access points and devices directly over UPnP as specified in the *WFADevice* and the *WFAWLANConfig* service specifications and also for configuring devices via a UPnP proxy function, typically implemented in an access point.

Registration Message Attributes

The following tables list the attributes for registration message exchange that Windows Vista uses. Devices may use additional messages (such as Beacon and Probe Response) for exchange over the wireless network, but Windows Vista does not provide or consume these messages. Message exchange is done as defined in the UPnP *WFADevice* specification.

Table 3. Probe Request

Attribute	R/O	Notes
Version	R	0x10 = version 1.0, 0x11 = version 1.1, and so on.
Request Type	R	
Config Methods	R	
UUID-E	R	
Primary Device Type	R	
RF Bands	R	
Association State	R	

Attribute	R/O	Notes
Configuration Error	R	
Device Password ID	R	

Table 4. M1

Attribute	R/O	Notes
Version	R	0x10 = version 1.0, 0x11 = version 1.1, and so on.
Message Type	R	Value is 0x04 for M1.
UUID-E	R	
MAC Address	R	
Enrollee Nonce	R	
Public Key	R	Diffie-Hellman key of enrollee. Key size and group are implied by the attribute data size.
Authentication Type Flags	R	
Encryption Type Flags	R	
Connection Type Flags	R	
Config Methods	R	
Simple Config State	R	
Manufacturer	R	
Model Name	R	
Model Number	R	
Serial Number	R	
Primary Device Type	R	
Device Name	R	
RF Bands	R	Specific RF band used for this message
Association State	R	
Device Password ID	R	
Configuration Error	R	
OS Version	R	
Feature ID	O	

Table 5. M2

Attribute	R/O	Notes
Version	R	0x10 = version 1.0, 0x11 = version 1.1, and so on.
Message Type	R	Value is 0x05 for M2.
Enrollee Nonce	R	
Registrar Nonce	R	
UUID-R	R	
Public Key	R	Diffie-Hellman key of registrar, key size, and group are implied by the attribute data size.
Authentication Type Flags	R	
Encryption Type Flags	R	
Connection Type Flags	R	
Config Methods	R	
Manufacturer	R	
Model Name	R	
Model Number	R	

Attribute	R/O	Notes
Serial Number	R	
Primary Device Type	R	
Device Name	R	
RF Bands	R	Specific RF band used for this message
Association State	R	
Configuration Error	R	
Device Password ID	R	The device password ID that the registrar indicates may be different from the ID that the enrollee in M1 sends.
OS Version	R	
Feature ID	O	
Authenticator	R	

Table 6. M2D

Attribute	R/O	Notes
Version	R	0x10 = version 1.0, 0x11 = version 1.1, and so on.
Message Type	R	Value is 0x06 for M2D.
…		Same as M2, except no **Public Key**, no **Encrypted Data**, and no **Authenticator** attribute.

Table 7. M3

Attribute	R/O	Notes
Version	R	0x10 = version 1.0, 0x11 = version 1.1, and so on.
Message Type	R	Value is 0x07 for M3.
Registrar Nonce	R	
E-Hash1	R	Hash of first half of device password, DH secret, and secret nonce 1.
E-Hash2	R	Hash of second half of device password, DH secret, and secret nonce 2.
Authenticator	R	

Table 8. M4

Attribute	R/O	Notes
Version	R	0x10 = version 1.0, 0x11 = version 1.1, and so on.
Message Type	R	Value is 0x08 for M4.
Enrollee Nonce	R	
R-Hash1	R	Hash of first half of device password, DH secret, and secret nonce 1.
R-Hash2	R	Hash of second half of device password, DH secret, and secret nonce 2.
Encrypted Settings	R	**Encrypted Secret Nonce** attribute that contains the registrar's secret nonce 1.
Authenticator	R	

Table 9. M5

Attribute	R/O	Notes
Version	R	0x10 = version 1.0, 0x11 = version 1.1, and so on.
Message Type	R	Value is 0x09 for M5.
Registrar Nonce	R	
Encrypted Settings	R	**Encrypted Secret Nonce** attribute that contains the enrollee's secret nonce 1.
Authenticator	R	

Table 10. M6

Attribute	R/O	Notes
Version	R	0x10 = version 1.0, 0x11 = version 1.1, and so on.
Message Type	R	Value is 0x0a for M6.
Enrollee Nonce	R	
Encrypted Settings	R	**Encrypted Secret Nonce** attribute that contains the registrar's secret nonce 2.
Authenticator	R	

Table 11. M7

Attribute	R/O	Notes
Version	R	0x10 = version 1.0, 0x11 = version 1.1, and so on.
Message Type	R	Value is 0x0b for M7.
Registrar Nonce	R	
Encrypted Settings	R	**Encrypted Secret Nonce** attribute that contains the enrollee's secret nonce 2 and current wireless settings if the enrollee is an access point.
Authenticator	R	

Table 12. M8

Attribute	R/O	Notes
Version	R	0x10 = version 1.0, 0x11 = version 1.1, and so on.
Message Type	R	Value is 0x0c for M8.
Enrollee Nonce	R	
Encrypted Settings	R	Encrypted wireless settings for enrollee. This attribute may also include a digital certificate.
Authenticator	R	

The **Encrypted Setttings** attribute in M8 that is sent to access points or stations may contain multiple network keys and associated binding information (SSID, MAC Address, Authentication Type and Encryption Type), but the Windows Vista release to manufacture (RTM) will send only a single network key and associated binding information to access points and likewise, a single **Credential** to stations.

Table 13. Encrypted Settings Attribute in M8 for Access Point

Attribute	R/O	Notes
Network Index	O	This attribute is used only if the enrollee is an access point and the registrar wants to configure settings for a nondefault network interface. If omitted, the **Network Index** defaults to 1.
SSID	R	
Authentication Type	R	

Attribute	R/O	Notes
Encryption Type	R	
Network Key Index	O	If omitted, the **Network Key Index** defaults to 1.
Network Key	R	Multiple instances of **Network Key** and its preceding **Network Key Index** may be included.
MAC Address	R	
Key Wrap Authenticator	R	

Table 14. Encrypted Attributes in M8 for Station

Attribute	R/O	Notes
Credential	R	May include multiple instances of **Credential**.
Key Wrap Authenticator	R	

Table 16. WCN-NET_Ack

Attribute	R/O	Notes
Version	R	0x10 = version 1.0, 0x11 = version 1.1, and so on.
Message Type	R	Value is 0xD for WCN-NET_ACK message.
Enrollee Nonce	R	
Registrar Nonce	R	

Table 17. WCN-NET_Nack

Attribute	R/O	Notes
Version	R	0x10 = version 1.0, 0x11 = version 1.1, and so on.
Message Type	R	Value is 0xE for WCN-NET_NACK message.
Enrollee Nonce	R	
Registrar Nonce	R	
Configuration Error	R	

Table 18. WCN-NET_Done

Attribute	R/O	Notes
Version	R	0x10 = version 1.0, 0x11 = version 1.1, and so on.
Message Type	R	Value is 0xF for WCN-NET Done message.
Enrollee Nonce	R	
Registrar Nonce	R	

Transportation of Registration Protocol Messages

UPnP Transport

Access points that implement the Wi-Fi Simple Configuration Protocol operate as an enrollee by using UPnP when interacting with Windows Vista. The *WFADevice* and *WFAWLANConfig* service documents specify UPnP behavior for Ethernet-connected enrollees.

Wi-Fi Simple Configuration–capable enrollees can exchange Registration Protocol messages with Windows Vista either directly over UPnP or via a proxy function in a Wi-Fi Simple Configuration–capable access point. The *WFADevice* and *WFAWLANConfig* service documents specify UPnP behavior for Ethernet-connected enrollees. Note that when connected to Ethernet and advertising the

Wi-Fi Simple Configuration–capable UPnP device and service, enrollees must turn off wireless-based discovery. A device must not simultaneously advertise itself over UPnP and 802.11.

If an enrollee is using its Wi-Fi interface for running the Registration Protocol, it should continuously scan all available wireless networks with Probe request messages.

EAP Transport of Registration Protocol

WCN-NET uses 802.1X and EAP to transport in-band Registration Protocol messages. This protocol is mapped onto a custom EAP method that is described later in this specification. WCN-NET does not require the access point to support RADIUS, and the network is not required to include an authentication server. In fact, many WCN-NET–capable access points may support 802.1X only to configure WPA-Personal Credentials via WCN-NET. Enrollees that use WCN-NET are not granted direct access to the WLAN through the WCN-NET custom EAP method.

The EAP method is used to configure the enrollee with a credential that can be used subsequently with whatever access method that WLAN supports. For example, if the access point supports only WPA-Personal with a network-wide shared PSK, then the enrollee would run the WCN-NET EAP method to obtain the PSK, disassociate, and then reconnect and use WPA-Personal to access the WLAN. Alternatively, if the access point supports 802.1X authentication, the enrollee may first run the WCN-NET EAP method to obtain a shared secret credential and then reconnect by using that secret in conjunction with another EAP method to access the WLAN.

The WCN-NET EAP method can be used for discovering a registrar or enrollee or for establishing a credential. The first time that the enrollee encounters a new WLAN, it sends out its discovery information and executes the EAP method with each access point it finds that supports WCN-NET. In both the discovery message and in M1, the enrollee provides information about itself to the WLAN. The M2 and M2D messages sent to the enrollee likewise provide information about the available registrars. When the enrollee first discovers and attempts to connect to the WLAN, the WLAN's registrars may not yet know the enrollee's device password. Therefore, registrars without the device password respond with M2D messages.

Although these M2D messages are unauthenticated, they can help enrollees with rich user interfaces to guide the user through the enrollment process and can also help a headless enrollee select a particular registrar that may support optional or vendor-extended functions.

As the enrollee scans the M2D messages that registrars in the network sent, it may discover that none of them possesses its device password. Therefore, the enrollee can prompt the user to perform a trust bootstrapping operation such as connecting an available out-of-band channel or entering a device password into one of the available registrars. If the user decides to enter the enrollee's device password into the registrar, the enrollee discovers this the next time it connects and reruns the EAP method. It can then perform the complete Registration Protocol.

If the enrollee has no user interface to lead the user through the enrollment, it is likely that one or more of the WLAN's registrars can do this. Both the registrar and the enrollee are given sufficient information about each other's capabilities through the EAP method to successfully lead the user through the enrollment. If the user decides to use an out-of-band channel for registration, then M2 is implicitly authenticated by the channel and can carry the network configuration data. An enrollee with a limited user interface should continue to scan the available WLANs

for a registrar that returns M2 or sees a selected registrar flag in an access point's Beacon.

EAP Message Framing

The access point functions as the EAP authenticator on the WLAN. Thus, the access point generates EAP request messages, and enrollees and registrars generate EAP responses. If the registrar is external to the access point, then it uses UPnP (rather than RADIUS) to exchange Registration Protocol messages with the access point. A registrar may also function in the role of an 802.1X authenticator in ad-hoc mode. This latter mode is useful for networks with legacy access points.

The following section contains a brief summary of the WCN-NET EAP method. Figure 3 shows the EAP packet format for request and response messages. For a more complete discussion of these fields and EAP, refer to RFC 3748.

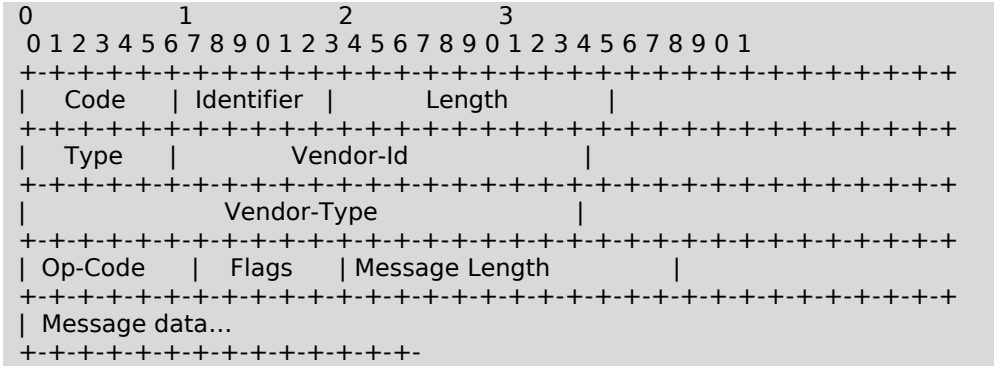


Figure 3: EAP Packet Format

The Code field is set to 1 for EAP-Request messages and to 2 for EAP-Response messages. The Identifier field is used to correlate request and response messages. The Length field gives the overall length of the EAP packet. The Type field indicates the EAP method type. For WCN-NET, it is set to 254 (expanded type).

The Vendor-Id is the WFA SMI code 0x00372A, and the Vendor-Type is 0x0000 0001 (SimpleConfig).

The Op-Code field is one of the following values:

0x01 : WCN-NET_Start

0x02 : WCN-NET_ACK

0x03 : WCN-NET_NACK

0x04 : WCN-NET_MSG

0x05 : WCN-NET_Done

0x06: WCN-NET_FRAG_ACK

The sequence of the messages that correspond to these Op-Code values is defined by the appropriate state machine that is associated with the scenario (adding an enrollee or adding an external registrar).

EAP Message Fragmentation and Reassembly

The Flags field is a bit-wise OR of flags.

0x01 : More fragments (MF)

0x02 : Length field (LF)

0x04 – 0x80 : reserved

If the MF flag is set, the original packet-required fragmentation and additional fragments still must be transmitted. The MF flag is not set if no additional packet fragments are expected. After receiving each packet with MF set, the receiving party responds with a WCN-NET_FRAG_ACK message. To reassemble the original packet, the receiving party concatenates the MessageData parts of each fragment.

If the LF flag is set, the Message Length field is included in the header to indicate the number of bytes of the entire message data being conveyed. If the LF flag is not set, then the Message Length field is omitted. The LF flag and Message Length field are included in the first EAP packet only for a fragmented EAP message. The LF flag must not be set for later fragments.

EAP fragmentation is specific to the EAP connection. If a message is fragmented for transmission over EAP, the supplicant and authenticator must handle fragmentation and reassembly of the frame. The proxy function that is required in access points must provide a completely assembled message to the UPnP interface.

EAP Identity

If the supplicant intends to add itself as an external registrar, it must use the EAP Identity “WFA-SimpleConfig-Registrar-1-0”. If it intends to acquire WLAN credentials as an enrollee, it must use the EAP Identity “WFA-SimpleConfig-Enrollee-1-0”.

EAP Messages

WCN-NET_Start

WCN-NET_Start is sent by the access point when it receives an EAP Response/Identity that contains the NAI “WFA-SimpleConfig-Enrollee-1-0”. The Message Data field of this message is empty.

WCN-NET_ACK

WCN-NET_ACK is sent by the supplicant or the authenticator when it successfully processes a message but does not have a message to send in response. For example, **WCN-NET_ACK** is sent by either the supplicant or authenticator when it has processed a message fragment and is ready for the next fragment. **WCN-NET_ACK** is also sent in response to M2D messages.

WCN-NET_NACK

WCN-NET_NACK is sent by the supplicant or the authenticator if it encounters an error when it authenticates or processes a message. If the supplicant is an enrollee, then this message is sent by the access point to all external registrars via a UPnP event. The Message Data field of this message is specified in “EAP Message Framing” earlier in this specification.

WCN-NET_MSG

WCN-NET_MSG may be sent by the supplicant or authenticator. Its **MessageData** payload contains a Registration Protocol message. The authenticator state machine does not examine these messages to determine their contents. It simply passes them along to the registrar or enrollee.

WCN-NET_Done

WCN-NET_Done is sent by the enrollee after it has successfully processed a **WCN-NET_M8** message. It indicates that the enrollee believes it has correctly received a credential for the WLAN. The Message Data field of this message shown in the **Done** message in Table 18.

WCN-NET_FRAG_ACK

WCN-NET_FRAG_ACK is sent by the supplicant or the authenticator when it successfully processes a fragmented EAP message and is ready for the next fragment.

Device Requirements

See the Windows Vista Logo Program requirements on the WHDC Web site.

Resources

E-mail:

rally@microsoft.com

Code Coverage Tools

From Intel: http://cache-www.intel.com/cd/00/00/21/92/219280_compiler_code-coverage.pdf

Batch command files to invoke Build on MSDN: http://g.msn.com/9SE/1? http://msdn2.microsoft.com/en-us/library/hefydhhy.aspx&&DI=6066&IG=94c2612141b9477096df2027e35c022e&POS=3&CM=WPU&CE=3&CS=AWP&SR=3

Debugging Tools for Windows and Windows Symbols

WinDbg and other kernel debuggers, extensions, and tools
http://www.microsoft.com/whdc/DevTools/Debugging/default.mspx

Microsoft Developer Network

Checked builds of Windows and other developer resources
http://msdn.microsoft.com

Verbose Debug Tracing - Microsoft Knowledge Base

How to enable verbose debug tracing in various drivers and subsystems
http://support.microsoft.com/default.aspx?scid=kb;en-us;314743

www.ietf.org/rfc/rfc3748.txt - RFC 3748

http://www.ietf.org/rfc/rfc3748.txt

UPnP Web Resources

http://www.upnp.org

Wi-Fi Alliance Certification (and WFADevice and WFAWLANConfig service documents)

http://www.Wi-Fi.org

Windows Driver Kit

http://www.microsoft.com/whdc/driver/WDK/aboutWDK.mspx

Windows Vista Logo Program

http://www.microsoft.com/whdc/winlogo/hwrequirements.mspx

Appendix A

Master Table—Data Component Set

The following tables enumerate the various attribute types that are defined for WCN-NET and used in Windows Vista. The sizes given in the Length column correspond to the Data part of the attribute. The overall size occupied by each attribute includes an additional 4 bytes (2 bytes of ID and 2 bytes of Length).

Table A1. Attribute Types Defined for WCN-NET

Description	ID (Type)	Length
802.1X Enabled	0x1062	Bool
AP Setup Locked	0x1057	1B
Application Extension	0x1058	<= 512B
AppSessionKey	0x1063	<= 128B
Association State	0x1002	2B
Authentication Type	0x1003	2B
Authentication Type Flags	0x1004	2B
Authenticator	0x1005	8B
Config Methods	0x1008	2B
Configuration Error	0x1009	2B
Connection Type	0X100C	1B
Connection Type Flags	0X100D	1B
Credential	0X100E	
Device Name	0x1011	<= 32B
Device Password ID	0x1012	2B
EAP Type	0x1059	<=8B
E-Hash1	0x1014	32B
E-Hash2	0x1015	32B
E-SNonce1	0x1016	16B
E-SNonce2	0x1017	16B
Encrypted Settings	0x1018	
Encryption Type	0X100F	2B
Encryption Type Flags	0x1010	2B
Enrollee Nonce	0x101A	16B
Feature ID	0x101B	4B
Initialization Vector	0x1060	32B
Key Identifier	0x101F	16B
Key Provided Automatically	0x1061	Bool
Key Wrap Authenticator	0X101E	8B
MAC Address	0x1020	6B
Manufacturer	0x1021	<= 64B
Message Counter	0x104E	8B
Message Type	0x1022	1B
Model Name	0x1023	<= 32B
Model Number	0x1024	<= 32B
Network Index	0x1026	1B
Network Key	0x1027	<= 64B

Description	ID (Type)	Length
Network Key Index	0x1028	1B
OS Version	0X102D	4B
Primary Device Type	0x1054	8B
Public Key	0x1032	192B
R-Hash1	0X103D	32B
R-Hash2	0X103E	32B
R-SNonce1	0X103F	16B
R-SNonce2	0x1040	16B
Registrar Nonce	0x1039	16B
Request Type	0x103A	1B
Response Type	0x103B	1B
RF Bands	0x103C	1B
Secondary Device Type List	0x1055	<= 128B
Selected Registrar	0x1041	Bool
Serial Number	0x1042	<= 32B
Simple Config State	0x1044	1B
SSID	0x1045	<= 32B
UUID-E	0x1047	16B
UUID-R	0x1048	16B
Version	0x104A	1B (int)
WEPTransmitKey	0x1064	1B
<Reserved for WFA>	0x1065 – 0x1FFF	
<Unavailable>	0x000 – 0x0FFF, 0x2000 – 0xFFFF	

Master Table Definitions

802.1X Enabled

This variable specifies if the network uses 802.1X for network authentication.

AP Setup Locked

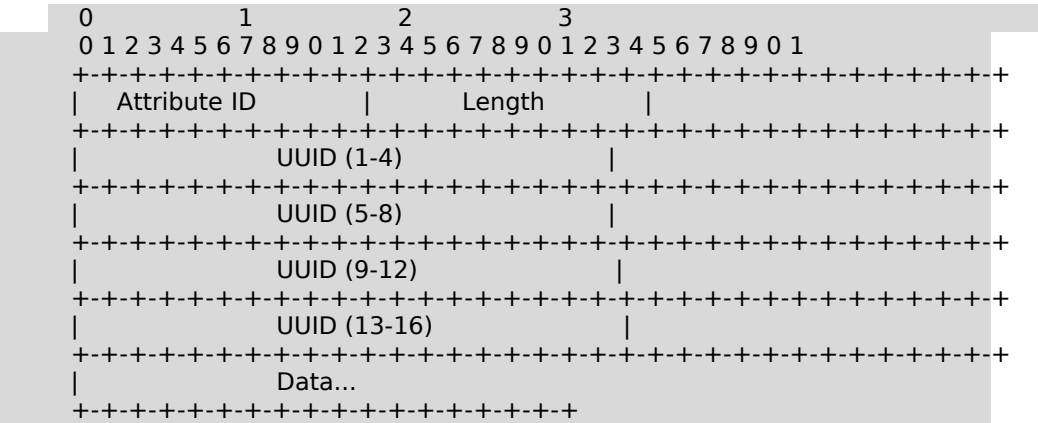
This variable indicates that the access point has entered a state in which it refuses to allow an external registrar to attempt to run the Registration Protocol by using the access point's PIN (with the access point acting as enrollee). The access point should enter this state if it believes a brute force attack is underway against the access point's PIN. When in this state, the access point *must* continue to allow other enrollees to connect and run the Registration Protocol with any external registrars or the access point's built-in registrar (if any). Only the use of the access point's PIN for adding external registrars is disabled in this state. The **AP Setup Locked** state can be reset to FALSE through an authenticated call to **SetAPSettings**. Access points may provide other implementation-specific methods of resetting the **AP Setup Locked** state as well.

Application Extension

The Application Extension attribute is used to pass parameters for enabling applications during the WSC exchange. It is similar to the Vendor Extension attribute except that instead of a 3-byte Vendor ID prefix to the Vendor Data field, a 16-byte UUID (as defined in RFC 4122) is used. This provides a virtually unlimited application ID space with a regular structure that can be easily mapped onto a generic application extension API. Furthermore, the 16-byte UUID value can be used to derive application-specific AMSKs as described in Section 6.3 or pass any necessary keying directly.

The enrollee may, for example, send two Application Extension attributes to the registrar in the Encrypted Settings of M7, one with UUID-A and one with UUID-X. If the registrar supports the application that corresponds to UUID-X but not UUID-A, the registrar may indicate to the enrollee that it also supports application X by sending an Application Extension with UUID-X in the Encrypted Settings of M8. Given this exchange, the enrollee and registrar can exchange application-specific information in the Data field such as application-specific keying or they can derive an AMSK for application X as follows:

AMSK = kdf(EMSK, N1 || N2 || UUID-X, 256)



One use of this Application Extension mechanism is to permit a Simple Config exchange to simultaneously set up connections for multiple wireless technologies (such as Wi-Fi or Bluetooth). To accomplish this setup, each network type would specify a UUID value for this purpose and define a corresponding Data element (to exchange data such as the device's MAC address on the other network). A network setup application on each device would exchange the Application Extension data by using the Simple Config Registration Protocol and then set up the other network connections by using that data with the native pairing mechanisms of the other networks.

Furthermore, if device pairing takes place first with another network type, it is possible to use the other network pairing mechanism as an out-of-band channel comparable to UFD or NFC. If this is done, the UUID and Data value to use for Simple Config are:

UUID=0xA6F6D81FB26941e2A72EC0B702248E90

Data=TLV attribute list below:

Attribute	R/O	Notes
Version	R	As defined in section 11.
OOB Device Password	O	May be omitted if OOB Device Password has already been received from peer device.
SSID	O	Included if SSID is known by sender.
<other…>	O	

Note that this approach passes the OOB Device Password directly in the Data field or can be used to pass transport specific parameters and keying directly to eliminate the requirement to rerun the Registration Protocol a second time.

AppSessionKey

The **AppSessionKey** attribute allows the exchange of application-specific session keys and may be used as an alternative to calculating AMSKs.

Association State

This component shows the configuration and association state that the wireless station is in when sending out a discovery request.

Table A2. Association State Values

Value	Description
0	Not associated
1	Connection success
2	Configuration failure
3	Association failure
4	IP failure

Authentication Type

This variable contains a specific value from the authentication types table for the enrollee (access point or station) to use.

Authentication Type Flags

This variable indicates the network authentication capabilities of the enrollee (access point or station). It provides a bitwise OR of the fields in the following table.

Table A3. Authentication Type Values

Value	Description
0x0001	Open
0x0002	WPAPSK
0x0004	Shared
0x0008	WPA
0x0010	WPA2
0x0020	WPA2PSK

Authenticator

This component is a keyed hash of data. The specific data in the hash calculation depends upon the processing context. The hash algorithm for Easy Setup version 1.0 is HMAC-SHA-256. In the context of the Registration Protocol, the default key that is used in the HMAC is **AuthKey**. If a nondefault key is used, the key is specified in the **Key Identifier** attribute immediately preceding the **Authenticator** attribute. To reduce message payload size, the **Authenticator** attribute's **Data** component includes only the first 64 bits of the HMAC-SHA-256 output.

Config Methods

This component lists the configuration methods that the enrollee or registrar supports. The list is a bitwise OR of values from the following table. In addition to **Config Methods**, access points and stations that support the UPnP Management Interface must support the permitted **Config Methods** attribute, which is used to control the **Config Methods** that are enabled on that access point.

Table A4. Config Methods Values

Value	Hardware interface
0x0001	USBA (flash drive)
0x0002	Ethernet
0x0004	Label
0x0008	Display
0x0010	External NFC token
0x0020	Integrated NFC token
0x0040	NF interface
0x0080	PushButton
0x0100	Keypad

Configuration Error

This component shows the result of the device attempting to configure itself and associate with the WLAN.

Table A5. Configuration Errors Values

Value	Description
0	No error
1	OOB interface read error
2	Decryption CRC failure
3	2.4 channel not supported
4	5.0 channel not supported
5	Signal too weak
6	Network authentication failure
7	Network association failure
8	No DHCP response
9	Failed DHCP configuration
10	IP address conflict
11	Could not connect to registrar
12	Multiple PBC sessions detected
13	Rogue activity suspected
14	Device busy
15	Setup locked

Value	Description
16	Message timeout
17	Registration session timeout
18	Device password authentication failure

The device busy error is returned if the sending device cannot respond to the request due to some internal conflict or resource contention issue. For example, if a device can perform only a single instance of the Registration Protocol, it may return this error in response to attempts to start another instance in the middle of an active session.

Connection Type

This attribute contains a specific value from the connection type flags table for the enrollee (access point or station) to use.

Connection Type Flags

This variable represents the capabilities of the enrollee.

Table A6. Connection Type Flags Values

Value	Description	Required/optional
0x1	ESS	R
0x2	IBSS	R

Credential

This compound attribute contains a single WLAN credential. The subattributes in credential are shown in the following table.

Table A7. Credential Attributes

Attribute	R/O	Notes, allowed values
Network Index	R	
SSID	R	SSID of access point or ad-hoc network
Authentication Type	R	
Encryption Type	R	
Network Key Index	R	
Network Key	R	
MAC Address	R	Member device’s MAC address
EAP Type	O	
EAP Identity	O	
Key Provided Automatically	O	
802.1X Enabled	O	

Device Name

This component is a user-friendly description of the device encoded in UTF-8. Typically, this would be a unique identifier that describes the product in a way that is recognizable to the user. The device must have this field populated from the factory, and it is recommended that it indicate the manufacturer and model.

Device Password ID

This attribute is used to identify a device password. There are six predefined values and ten reserved values. If the Device Password ID is Default, then the enrollee should use its PIN password (from the label or display). This password may correspond to the label, display, or a user-defined password that has been configured to replace the original device password.

"User-specified" indicates that the user has overridden the password with a manually selected value. "Machine-specified" indicates that the original PIN password has been overridden by a strong, machine-generated device password value. The Rekey value indicates that the device's 256-bit rekeying password will be used. The PushButton value indicates that the PIN is the all-zero value that was reserved for the PushButton Configuration method.

The registrar-specified value indicates a PIN that has been obtained from the registrar (via a display or other out-of-band method). This value may be further augmented with the optional "Identity" attribute in M1. This augmentation is useful when multiple predefined UserID/PIN pairs have been established by a registrar such as an authenticator used for Hotspot access. If the Device Password ID in M1 is not one of the predefined or reserved values, it corresponds to a password that was given to the registrar as an OOB Device Password.

Table A8. Device Password ID Values

Value	Description
0x0000	Default (PIN)
0x0001	User-specified
0x0002	Machine-specified
0x0003	Rekey
0x0004	PushButton
0x0005	Registrar-specified
0x0006 – 0x000F	Reserved

EAP Type

This attribute contains the binary representation of an EAP type as found in an EAP packet. If it is a standard EAP type, it is only a single byte. Extended EAP types, such as the Wi-Fi Simple Config Registration Protocol (refer to EAP transport section), may be up to 8 bytes (1-byte Type, 3-byte Vendor-ID, and 4-byte Vendor-Type).

E-Hash1

This component is the HMAC-SHA-256 hash of the first half of the device password and the enrollee's first secret nonce.

E-Hash2

This component is the HMAC-SHA-256 hash of the second half of the device password and the enrollee's second secret nonce.

E-SNonce1

This component is the first nonce that the enrollee uses with the first half of the device password.

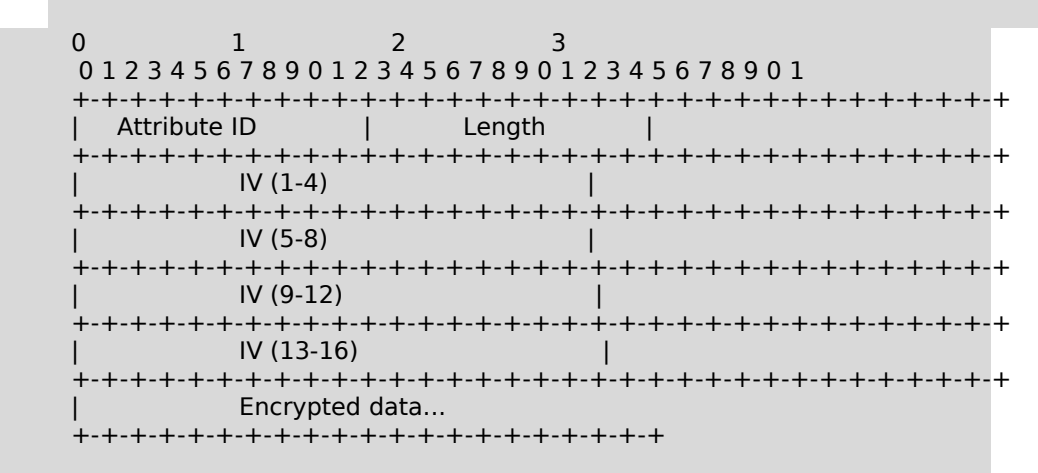
E-SNonce2

This component is the second nonce that the enrollee uses with the second half of the device password.

Encrypted Settings

The Data field of the **Encrypted Settings** attribute includes an initialization vector (IV) followed by a set of encrypted **Simple Config TLV** attributes. The last attribute in the encrypted set is a key wrap authenticator computed according to the procedure described in "Master Table Definitions." In the context of the Registration Protocol, the default key that is used for the encryption is the **KeyWrapKey**.

The encryption algorithm is AES in CBC mode. In other contexts, the key is specified in a **Key Identifier** attribute that immediately precedes the **Encrypted Settings** attribute. The data structure of the **Encrypted Settings** attribute is as follows.



If an alternative key wrap algorithm is later preferred, it can be added by defining a new attribute with a different attribute ID. The key wrap authenticator is 96 bits long (32 bits of attribute ID and length, and 64 bits of the HMAC-SHA-256 output). This implies that the total overhead for an **Encrypted Settings** attribute is 256 bits (32 bits of attribute ID and length, 128 bits of IV, and 96 bits of key wrap authenticator).

Encryption Type

This attribute contains a specific value from the encryption type flags table for the enrollee (access point or station) to use.

Encryption Type Flags

This attribute is a binary OR set of WLAN encryption types that are supported by the enrollee (one or more from the following table).

Table A9. Encryption Type Flags Value

Value	Description
0x0001	None
0x0002	WEP
0x0004	TKIP
0x0008	AES

Enrollee Nonce

This component is a randomly generated binary value that the enrollee creates for setup.

Feature ID

This attribute indicates a particular feature build for an operating system that is running on the device. It is a 4-byte field, with the most significant bit reserved and always set to 1.

Initialization Vendor

A randomly generated block of bits that is combined with the data to be encrypted to provide entropy to the encryption process.

Key Identifier
This attribute contains a 128-bit key identifier. If this attribute immediately precedes an **Encrypted Data** or **Authenticator** attribute, then the key that corresponds to the 128-bit identifier should be used to decrypt or verify the Data field.

Key Provided Automatically
This variable specifies whether the network provides the key.

Key Wrap Authenticator
This attribute contains the first 64 bits of the HMAC-SHA-256 computed over the data to be encrypted with the key wrap algorithm. It is appended to the end of the **ConfigData** before encryption, as described earlier in the "Registration Protocol" section.

MAC Address
This attribute is a 6-byte value that contains the 48-bit value of the MAC address. An example is 0x00 0x07 0xE9 0x4C 0xA8 0x1C.

Manufacturer
This component is an ASCII string that identifies the manufacturer of the device. Generally, this field should allow a user to make an association with a device with the labeling on the device.

Message Counter
This variable contains a 64-bit counter that is included in certain messages to prevent replay attacks. It is not required in Registration Protocol messages, but it is used in many of the UPnP-based management interface messages.

Message Type
This variable identifies the specific message being sent by the enrollee or registrar, as shown in the following table.

Table A10. Message Type Values

Value	Description
0x01	Beacon
0x02	Probe request
0x03	Probe response
0x04	M1
0x05	M2
0x06	M2D
0x07	M3
0x08	M4
0x09	M5
0x0A	M6
0x0B	M7
0x0C	M8
0x0D	WCN-NET_ACK
0x0E	WCN-NET_NACK
0x0F	WCN-NET_DONE

Model Name
This attribute is an ASCII string that identifies the model of the device. Generally, this field should allow a user to make an association with a device with the labeling on the device.

Model Number

This attribute provides additional description of the device to the user.

Network Index

This variable is used to get and set network settings for devices that host more than one network. The default value is 1 and refers to the primary WLAN network on the device.

Network Key

This variable specifies the wireless encryption key for the enrollee to use. This field is interpreted as shown in the following table.

Table A11. Network Key

Authentication	Encryption	Network key type
None	None	0 ASCII characters
WPAPSK (Passphrase)	TKIP/AES	8–63 ASCII characters
WPAPSK	TKIP/AES	64 Hex characters
Shared/Open	WEP	5 or 13 ASCII characters 10 or 26 Hex characters

Network Key Index

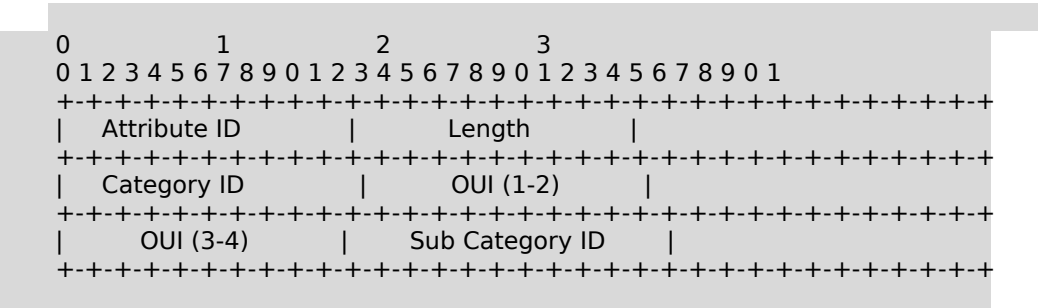
This variable specifies a particular **Network Key** instance.

OS Version

This component indicates what operating system is running on the device. It is a 4-byte field, with the most significant bit reserved and always set to 1.

Primary Device Type

This attribute contains the primary type of the device. This following shows its format:



Vendor-specific subcategories are designated by setting the OUI to the value that is associated with that vendor. Note that a 4-byte subdivided OUI is used. For the predefined values, the Wi-Fi Alliance OUI of 00 50 F2 04 is used. The predefined values for category ID and subcategory ID are as shown in the following table. Note that there is no way to indicate a vendor-specific main device category. The OUI applies only to the interpretation of the subcategory.

Table A12. Category ID and Subcategory ID Values

Category	ID value	Subcategory	ID value
Computer	1	PC	1
		Server	2
		Media Center	3
Input Device	2		
Printers, Scanners, Faxes, and Copiers	3	Printer	1
		Scanner	2
Camera	4	Digital Still Camera	1

Category	ID value	Subcategory	ID value
Storage	5	NAS	1
Network Infrastructure	6	Access point	1
		Router	2
		Switch	3
Displays	7	Television	1
		Electronic Picture Frame	2
		Projector	3
Multimedia Devices	8	DAR	1
		PVR	2
		MCX	3
		DMR	4
Gaming Devices	9	Xbox	1
		Xbox360	2
		Playstation	3
Telephone	10	Windows Mobile	1

Public Key

This variable represents the sender's Diffie-Hellman public key. The length of the attribute indicates the size of the key as well as the specific generator and prime.

R-Hash1

This variable is the HMAC-SHA-256 hash of the first half of the device password and the registrar's first secret nonce.

R-Hash2

This variable is the HMAC-SHA-256 hash of the second half of the device password and the registrar's second secret nonce.

R-SNonce1

This variable is the first nonce that the registrar uses with the first half of the device password.

R-SNonce2

This variable is the second nonce that the registrar uses with the second half of the device password.

Registrar Nonce

This component is a randomly generated binary value that the registrar creates for setup.

Request Type

This component specifies the mode in which the device operates for this setup exchange. If the device is an enrollee, it may send out only discovery messages or may also request that the registrar proceed with opening a data connection. This allows enrollees to more efficiently discover devices on the network. If the device indicates that it wants to engage setup as a registrar, the access point must indicate that it will operate as an access point in the response. The **Request Type** attribute is carried throughout the 802.1X data channel setup process in the **Simple Config** IE. There are two subtypes of registrars:

- A WLAN manager registrar indicates that this registrar intends to manage the access point or station settings by using UPnP. This means it will derive a UPnP AP or STA management key.

- The ordinary registrar type indicates that this registrar does not intend to subsequently manage the enrollee's settings. Access points must not derive access point management keys for an ordinary registrar.

If a registrar does not need to be a WLAN manager registrar, it should set the **Request Type** to Registrar. This avoids needlessly consuming resources on the access point.

Table A13. Request Type Values

Value	Description
0x00	Enrollee, Info only
0x01	Enrollee, open 802.1X
0x02	Registrar
0x03	WLAN manager registrar

Response Type

This component specifies if the mode in which the device operates for this setup exchange. The **Response Type** IE is carried throughout the 802.1X data channel setup process.

Table A14. Response Type Values

Value	Description
0x00	Enrollee, Info only
0x01	Enrollee, open 802.1X
0x02	Registrar
0x03	Access Point

RF Bands

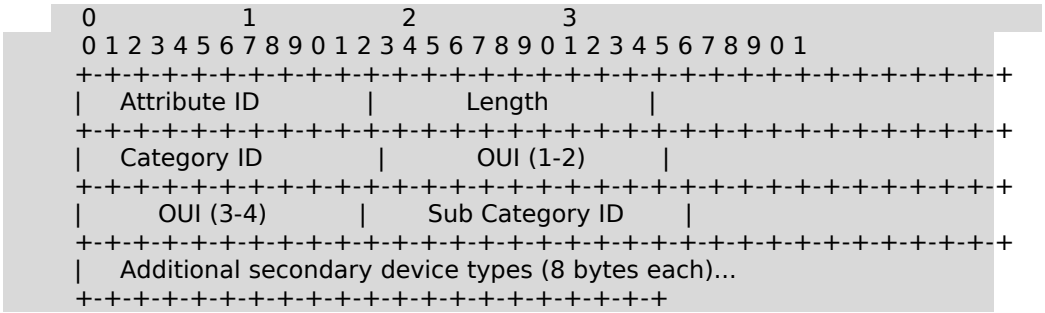
This attribute indicates a specific RF band that is used during message exchange to permit end points and proxies to communicate over a consistent radio interface.

Table A15. RF Bands Values

Value	Description
0x01	2.4GHz
0x02	5.0GHz

Secondary Device Type List

This attribute contains a list of secondary device types that the device supports. OUI and standard values for Category ID and Sub Category ID fields are defined in the Primary Device Type attribute. The Secondary Device Type List format is as follows:



Selected Registrar
This field indicates that a user has selected a registrar and that an enrollee should proceed with setting up an 802.1X uncontrolled data port with the registrar.

Serial Number
This component identifies the serial number of the enrollee.

Simple Config State
This variable indicates if a device has been previously configured (or settings actively accepted) by a user. A device's default factory behavior is "Not configured" and the device reverts to this setting when it has been reset to factory settings. After being successfully configured by a Windows Vista registrar, a device changes from "Not configured" to "Configured."

Table A16. Simple Config State Values

Value	Description
0x1	Not configured
0x2	Configured

SSID
This variable represents the service set identifier or network name. The client uses this to connect to the wireless network. This variable is read/write, and the field is always 32 bytes long.

UUID-E
The universally unique identifier (UUID) element is a unique globally unique identifier (GUID) that the enrollee generates. It uniquely identifies an operational device and should survive reboots and resets. The UUID is provided in binary format. If the device also supports UPnP, then the UUID corresponds to the UPnP UUID.

UUID-R
The UUID element is a unique GUID that the registrar generates. It uniquely identifies an operational device and should survive reboots and resets. The UUID is provided in binary format. If the device also supports UPnP, then the UUID corresponds to the UPnP UUID.

Version
This variable specifies the Easy Setup version. The 1-byte field is divided into a 4-bit major part that uses the top most significant bits (MSBs) and 4-bit minor part that uses the least significant bits (LSBs). As an example, version 3.2 would be 0x32.

WEPTransmitKey
This attribute identifies the Key Index that is used as the access point transmit key for WEP configurations.

Appendix B. *WFADevice:1* Device Template Version 1.01

For UPnP Version 1.0

January 23, 2006

Authors:
David Roberts, Microsoft Corporation
Victor Lortz, Intel Corporation

Notice

Contents

B.1 Overview and Scope

This device template complies with the UPnP Architecture, Version 1.0, as a vendor extended UPnP device.

This document defines the REQUIRED **ROOT** device:

urn:schemas-wifialliance-org:device:*WFADevice*

The *WFADevice* encapsulates services for the WCN-NET Device Control Protocol (DCP).

The Wi-Fi Alliance (WFA) device implements IP-based transports (such as Ethernet/802.3 wired standards) to provide an out-of-band mechanism to configure IEEE 802.11 (a, b, and g) settings on the device so the device may authenticate and associate with a secured wireless network and on access points so that the access point may host secure wireless networks.

B.1.1 Focus and Goals for DCP Version 1.0

WCN-NET has specific requirements for creating and extending wireless networks and adding wireless stations:

- Setup that is simple, a task that a typical consumer can complete
- Setup process that is secure and must maximize security of the WLAN
- A consumer focus with some attention to small business cases

B.1.2 Non-Goals for DCP Version 1.0

The following work items were considered to be beyond the scope of this version of the DCP.

- Replacing or enhancing the link security mechanism of the WLAN
- Configuration services for access points for "hotspot" and enterprise networks

B.1.3 WLAN Security Requirements and Recommendations

Link security is critical for wireless home network because connectivity is not restricted by the reach of wires or availability of physical ports. The likelihood of unintentional cross-links and malicious drive-by attacks will likely increase along with the popularity of WLANs. This will be detrimental to the user experience with wireless networks and will impede introduction of new product categories and usage models. Consumers and service providers will demand link security as part of the WLAN package.

An alternative to link security is to protect specific resources with security mechanisms involving higher (network or application) layers of the networking software stack. However, the average home user cannot be expected to be technically savvy and to take the time to identify all the vulnerable points (data/devices) in the home network for protecting himself or herself individually with appropriate methods.

Currently the most common way to secure 802.11 links in the home involves Wired-Equivalent Privacy (WEP)–based encryption and authentication. The security risks when using WEP are well known. An attacker can crack the WEP key by collecting packets with a wireless packet sniffer and running widely available utilities to determine the WEP key. If the WLAN owner becomes aware of the security compromise, the WEP key on all the clients and access point must be updated because the same WEP key is used for all nodes.

To build consumer confidence and expand usage of wireless applications, it is important for the home WLAN devices to adopt stronger security mechanisms such as Wi-Fi Protected Access (WPA) that has become available in the market. Longer term, it is expected that the security specification being worked on in the 802.11i working group would be the widely adopted and appropriate solution for a strong security mechanism on the access point. The security enhancements provide per-user based authentication, per-session keys, frequent rekeying, and stronger encryption methods such as Advanced Encryption Standard (AES).

One of the main issues with the use of security in WLAN is the process of setting up the security parameters. Current mechanisms for initializing link security on an access point device are not very user friendly. For example, with the WEP-based model, the user must retrieve a long WEP key for the access point either through a secure/wired connection first and enter it on the new client correctly. This problem of bootstrapping also exists with the mechanisms proposed as an improvement on the plain WEP-based security. Because of this, users are likely to not enable security in their network, leading to several vulnerabilities. The objective of the 802.11 security initialization mechanism using UPnP as proposed in this document is to reduce user involvement and introduce an intuitive usage model for users to take advantage of the higher level of security.

An overall security solution should protect the user from "man-in-the-middle" attacks by preventing the user's client from associating with an unfriendly access point and the user's access point from associating with a foreign client. It should prevent session-hijack attacks by making sure all messages between the access point and station are authenticated.

The objective of the DCP is to enable a secure WLAN solution with combined Wi-Fi and IP-connected devices that implement the required elements specified in the DCP. The following figure shows the major functional components of the *WFADevice*.

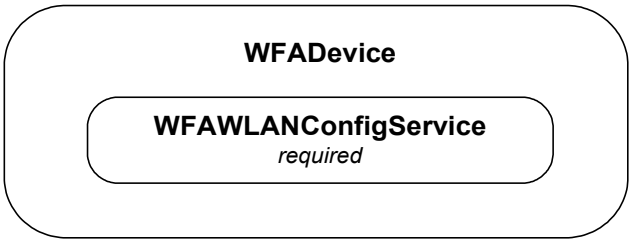


Figure B.1: Functional Components of *WFADevice*

B.1.3.1 Station Parameter Configuration

The *WFAWLANConfig* service, a required service of *WFADevice,* provides state variables for some of the wireless station parameters. They provide the ability to easily configure security and operation parameters, and offer diagnostic information. In addition, UPnP also provides event notification capability to inform clients that are responsible for Wi-Fi device configuration on the status of the Wi-Fi station.

The WCN-NET protocol that is imbedded in the exchange between the control point and the *WFADevice* perform authentication and encryption so that the UPnP actions do not require any additional security. The state variables in any *WFADevice* service must not return directly the binary data that is contained. If requested for a state variable action, the device may return a null or may choose to

follow the WCN-NET rules and encrypt the contents with an arbitrary SID or the SID of the requesting control point if known.

B.2 Device Definitions

B.2.1 Device Type

The following device type identifies a device that complies with this template:

urn:schemas-wifialliance-org:device:*WFADevice:1*

B.2.2 Device Model

WFADevice must be implemented with support for the WCN-NET secure association protocol.

B.2.2.1 Description of Device Requirements

The following table briefly describes the purpose of the services that are used in *WFADevice*.

Service name	Service description
WFAWLANConfig	Configuration parameters associated with a WLAN device that must be accessed programmatically.

Table B.1: Device Requirements for Stand-alone *WFADevice*

Device type	Root	Req. or opt.[1]	Service type	Req. or opt.[1]	Service ID[2]
			WFAWLANConfig:1	*R*	*WFAWLANConfig 1*
			Nonstandard services embedded by an UPnP vendor go here.	*X*	*TBD*

[1] R = Required, O = Optional, X = Nonstandard. .
[2] Prefixed by urn:wifialliance-org:serviceId: .

Figure B.2 shows the logical structure of the device and the single service defined for UPnP-enabled WLAN devices.

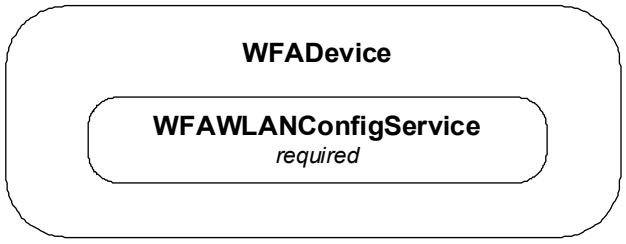


Figure B.2: *WFAWLANConfig Service* within *WFADevice*

B.2.3 Theory of Operation

This section describes the general usage model of the services defined in the WFA interface device. This section starts by listing the requirements and optional features of the WLAN nodes. This is followed by a section describing the various scenarios reflecting the use of these features. For each of these, the benefits that are enabled by the UPnP services are explicitly highlighted.

B.2.3.1 WLAN Node Requirements

From the perspective of requirements, a *WFADevice* is a WLAN node that also has an Ethernet or other established IP interface for, at a minimum, the purpose of discovering the device and configuring the WLAN settings of the device.

The WLAN node requirements are listed below:

- The device *must* be addressable via the Internet Protocol (IP) protocol by using the Ethernet interface.
- The device *must* provide a user the ability to physically reset it to factory default settings.
- The device *must* implement the WCN-NET protocol. This involves use of a predefined password (typically a PIN) and a public-private key pair and a cryptographic library for encryption and authentication.
- The device *may i*mplement the proxy function as defined in the *WFAWLANConfigService v1.0* specification. This service allows for any WLAN station to send an event when a new WLAN enrollee is requesting WLAN configuration. Note that the proxy function is required for access points.

B.2.3.1.1 Client Requirements for Configuration of WLAN Parameters

If configuration of WLAN parameters over the UPnP/IP channel is desired, at least one client in the LAN should have an interactive user interface. Other WLAN clients *may* be UPnP enabled or not, and they may be able to execute UPnP control point functionality to send UPnP actions to the WLAN station.

B.2.3.2 Configuration of New Clients to the WLAN

WFAWLANConfig service provides a set of actions to query and modify a set of 802.11 parameters on the WLAN client over an IP interface. Acquisition of the WLAN settings by the registrar performing configuration is performed in several ways:

- Using the actions defined in the *WFAWLANConfigService v1.0* or associated vendor extensions.
- From a local store on the registrar, possibly from a previous wireless client configuration for networks that use a single PSK.
- Current wireless settings of the registrar (for a WLAN attached device).
- Input from the user.

The PSK is encrypted by using the public key–based protocol that is specified the WCN-NET specification. Settings are sent by using the action specified in the *WFAWLANConfig* service. The data sets are validated and the settings are encrypted by using the WCN-NET exchange mechanism.

The *WFADevice* attempts to associate with the WLAN and reports state.

B.3 XML Device Description

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <URLBase>base URL for all relative URLs</URLBase>
  <device>
    <deviceType>urn:schemas-wifialliance-
org:device:WFADevice:1</deviceType>
    <friendlyName>short user-friendly title</friendlyName>
    <manufacturer>manufacturer name</manufacturer>
    <manufacturerURL>URL to manufacturer
site</manufacturerURL>
    <modelDescription>long user-friendly
title</modelDescription>
    <modelName>model name</modelName>
    <modelNumber>model number</modelNumber>
    <modelURL>URL to model site</modelURL>
    <serialNumber>manufacturer's serial number</serialNumber>
    <UDN>uuid:UUID</UDN>
    <UPC>Universal Product Code</UPC>
    <iconList>
      <icon>
        <mimetype>image/format</mimetype>
        <width>horizontal pixels</width>
        <height>vertical pixels</height>
        <depth>color depth</depth>
        <url>URL to icon</url>
      </icon>
      <!-- XML to declare other icons, if any, go here -->
    </iconList>
    <serviceList>
      <service>
        <serviceType>urn:schemas-wifialliance-
org:service:WFAWLANConfig:1</serviceType>
        <serviceId>urn:wifialliance-
org:serviceId:WFAWLANConfig1</serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
      </service>
            <!-- Declarations for other services added by
UPnP vendor (if any) go here -->
    </serviceList>
    <deviceList>
      Description of embedded devices added by UPnP vendor
(if any) go here
    </deviceList>
    <presentationURL>URL for presentation</presentationURL>
  </device>
</root>
```

B. 4 Test

No semantic tests are defined for this device.

Appendix C. *WFAWLANConfig:1* Service Template Version 1.01

For UPnP Version 1.0

Date: January 23, 2006

Authors:
David Roberts, Microsoft Corporation
Victor Lortz, Intel Corporation

Notice

Contents

C.1 Overview and Scope

This service definition complies with the UPnP Device Architecture, Version 1.0.

This service enables the configuration of IEEE 802.11 wireless stations and access points by using IP-based channels such as Ethernet and provides a proxy function between the IP network and 802.11 WCN-NET frames. It leverages the data structure and protocols defined for WCN-NET to reduce device complexity for supporting the UPnP methods.

This service-type enables the following functions:

- Device type and information.
- Proxy function on access points and optionally other devices to enable IP-based registrars to configure WLAN enrollees.
- Device information and capabilities.
- Secured exchange with UPnP devices for retrieving and configuring wireless parameters for access point management and wireless setup of Ethernet-attached stations.

C.2 Service Modeling Definitions

C.2.1 ServiceType

The service is required as specified in:

urn:schemas-wifialliance-org:device:*WFADevice:1*

The following service type identifies a service that complies with this template:

urn:schemas-wifialliance-org:*WFAWLANConfig:1*

This service does not support the **QueryStateVariable** action.

C.2.2 State Variables

Table C.1 shows all of the state variables of the *WFAWLANConfig* service. These variables are represented in the following order:

- The first set of variables is for message exchange.
- The second set shows the device capabilities message.
- The third set has the device configuration messages.
- The last set lists evented messages.

Table C.1: State Variables

Variable name	Req. or opt.[1]	Data type	Allowed value[1]	Default value[2]	Eng. units
Message	R	Bin.base64	See WCN-NET	N/A	N/A
InMessage	R	Bin.base64	See WCN-NET	N/A	N/A
OutMessage	R	Bin.base64	See WCN-NET	N/A	N/A
DeviceInfo	R	Bin.base64		N/A	
APSettings	C	Bin.base64		N/A	N/A
APStatus	C	ui1	See Table C.2	0	N/A
STASettings	C	Bin.base64		N/A	N/A
STAStatus	C	ui1	See Table C.2	0	N/A
WLANEvent	C	Bin.base64	See WCN-NET	N/A	
WLANEventType	C	ui1	See Table C.3	N/A	
WLANEventMAC	C	String	MAC Address, “xx:xx:xx:xx:xx:xx”, case-independent, 17 char	N/A	
Nonstandard state variables implemented by an UPnP vendor go here.	*X*	*TBD*	*TBD*	*TBD*	*TBD*

[1] R = Required, O = Optional, C = Conditional, X = Nonstandard

[2] Values listed in this column are required. To specify standard optional values or to delegate assignment of values to the vendor, reference a specific instance from the appropriate following table.

C.2.2.1 Message

This variable contains the TLV format for the data components passed in the message exchange between the registrar and the enrollee. The message identification (M2 through M8) is contained in the TLV binary format. Messages received from WLAN devices via Probe requests and WCN-NET 802.1X/EAP methods are evented.

C.2.2.2 InMessage

This variable contains the TLV format for the data components passed in the message exchange between the registrar and the enrollee. The message identification (M2-M8) is contained in the TLV binary format. Messages received from WLAN devices via Probe requests and WCN-NET 802.1X/EAP methods are evented. This provides separation of state variables for actions that use both input and output messages, such as **PutMessage**.

C.2.2.3 OutMessage

This variable contains the TLV format for the data components passed in the message exchange between the registrar and the enrollee. The message identification (M2-M8) is contained in the TLV binary format. Messages received from WLAN devices via Probe requests and WCN-NET 802.1X/EAP methods are evented. This provides separation of state variables for actions that use both input and output messages, such as **PutMessage**.

C.2.2.4 DeviceInfo

This variable contains the WCN-NET data components in TLV format that are provided in message M1. The allowed list is defined in the WCN-NET specification.

C.2.2.5 APSettings

This variable contains the TLV format for the data components passed in the management interface between the registrar and the access point and carries a message that is defined based on the action. This variable is required for access points.

APSettings are authenticated, and some parts are also encrypted. The symmetric keys for authentication and encryption are derived during the registrar configuration process. The **SID** attribute included in the messages enables the access point and registrar to determine which keys to use. The **APSettings** data components are transported in the **Encrypted Settings** data component value field.

The **APSettings** data components are described in the WCN-NET specification. Attributes retrieved by **GetAPSettings** are described as a **GetAPSettings** output message. Those set by **SetAPSettings**, are described as a **SetAPSettings** message. Those removed by **DelAPSettings** are described as a **DelAPSettings** message.

C.2.2.6 APStatus

This variable represents changes in the status of an access point. It is used in eventing to indicate changes on the access point that are related to configuration and attack mitigation. Table C.2 indicates the flag values of this field. It is required for access points.

Table C.2. Allowed Value List for APStatus

Value	Req. or opt.[1]	Description
0x00000001	R	Configuration change
0x00000010	R	Failed auth threshold reached (**APLocked**)

[1] *R = Required, O = Optional, C = Conditional, X = Nonstandard*

C.2.2.7 STASettings

This variable contains the TLV format for the data components passed in the configuration interface between the registrar and an established station on the WLAN and carries a message that is defined based on the action. This variable is required for 802.11 stations that support Ethernet configuration.

STASettings are encrypted and authenticated. The symmetric key is derived during the registrar configuration process and is referenced by using the **Key Identifier** data component. The **STASettings** data components are transported in the **Encrypted Settings** data component value field (preceded by a **Key Identifier**).

The **STASettings** data components are described in the WCN-NET specification in the section entitled "STASettingsMessage."

C.2.2.8 STAStatus

This variable represents changes in the status of a station. It is used in eventing to indicate changes on the station that are related to configuration and attack mitigation. Table C.2 indicates the flag values of this field. It is required for 802.11 stations.

C.2.2.9 WLANEvent

This variable represents the concatenation of the **WLANEventType**, **WLANEventMac**, and the 802.11 **WSC** message received from the enrollee and forwarded by the proxy over UPnP. It is required for access points and other proxies. **WLANEvent** is represented as base64 (WLANEventType || WLANEventMAC || Enrollee's message).

C.2.2.10 WLANEventType

This variable represents the type of **WLANEvent** frame that was received by the proxy on the 802.11 network. Table C.3 shows the options for this variable. This variable is required for access points and any other device that implements the proxy function.

Table C.3. Allowed Value List for WLANEventType

Value	Req. or opt.[1]	Description
1	R	802.11 WCN-NET Probe Frame
2	R	802.11 WCN-NET 802.1X/EAP Frame

[1] *R = Required, O = Optional, C = Conditional, X = Nonstandard*

C.2.2.11 WLANEventMAC

This variable represents the MAC address of the WLAN enrollee that generated the 802.11 frame that was received by the proxy. This variable is required for access points and any other device that implements the proxy function.

C.2.3. Eventing and Moderation

Table C.4 lists the events generated in this service where the first three are generated as part of a single propertyset and the remainder are independently generated. A single event is generated by each WCN-NET probe and 802.1X/WCN-NET EAP frame that is received on the 802.11 network. The remaining state variables are not evented.

***Table C.*4. Event Moderation**

Variable name	Evented	Moderated event	Max. event rate[1]	Logical combination	Min delta per event[2]
WLANEvent	Yes	Yes	.2	OR	
APStatus	Yes	Yes	.5	OR	
STAStatus	Yes	Yes	.5	OR	
Nonstandard state variables implemented by an UPnP vendor go here.	TBD	TBD	TBD	TBD	TBD

[1] Determined by N, where Rate = (Event)/(N secs).
[2] (N) * (allowedValueRange Step).

C.2.3.1. Event Model

Three state variables in the *WFAWLANConfig* service are evented:

- **WLANEvent**: This state variable event indicates that a WCN-NET Probe request or an 802.1X/WCN-NET EAP frame was received on the 802.11 network. The proxy service issues the event.
- **APStatus**: This state variable event indicates that the access point has either had a configuration change or that the access point has had too many failed authentications against its PIN/password and has locked itself out of enrollee mode.
- **STAStatus**: This state variable event indicates that the station has either had a configuration change or that the station has had too many failed authentications against its PIN/password and has locked itself out of enrollee mode.

The events are moderated.

C.2.4 Actions

Table C.5 lists the required and optional actions for the WFA device WFAWLANConfig service. This is followed by detailed information about these actions, including short descriptions of the actions, the effects of the actions on state variables, and error codes defined by the actions.

Table C.5. Actions

Name	Req. or opt.[1]
GetDeviceInfo	R
PutMessage	R
GetAPSettings	C
SetAPSettings	C
DelAPSettings	C
GetSTASettings	C
SetSTASettings	C
DelSTASettings	C
PutWLANResponse	C
SetSelectedRegistrar	C
RebootAP	O
ResetAP	C
RebootSTA	O
ResetSTA	C

[1] R = Required, O = Optional, C = Conditional.

C.2.4.1 GetDeviceInfo

This action retrieves the device's M1 data. It is required for access points and Ethernet-attached stations.

Arguments

Table C.6. Arguments for GetDeviceInfo

Argument	Direction	relatedStateVariable
NewDeviceInfo	Out	DeviceInfo

Dependency on State (if any)

Effect on State (if any)

Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.

C.2.4.2 PutMessage

This action continues the WCN-NET exchange over UPnP to permit establishment of a registrar with an access point and to configure an Ethernet (IP)-attached wireless station. A **PutMessage** is issued only after receiving M1 from a device from a **GetDeviceInfo**. **PutMessage** is used first to send M2 from the registrar to the device (for which a M3 response is expected). Additional round trips in the Registration Protocol are accomplished with subsequent calls to **PutMessage**.

Arguments

Table C.7 Arguments for PutMessage

Argument	Direction	relatedStateVariable
NewInMessage	In	InMessage
NewOutMessage	Out	OutMessage

Dependency on State (if any)

Effect on State (if any)

Errors

errorCode	ErrorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.

C.2.4.3 GetAPSettings

This action is used to directly retrieve settings from an access point for which the registrar already has established a secure association with. The input message is the **GetAPSettings** input message in the WCN-NET specification. The output message is the **GetAPSettings** output message in the WCN-NET specification. This action is required for access points.

Arguments

Table C.8. Arguments for GetAPSettings

Argument	Direction	relatedStateVariable
NewMessage	IN	Message
NewAPSettings	OUT	APSettings

Dependency on State (if any)

Effect on State (if any)

Errors

errorCode	ErrorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.

C.2.4.4 SetAPSettings

This action is used to set settings on an access point for which the registrar already has established a secure association with. The input message is the **SetAPSettings** message in the WCN-NET specification. This action is required for access points.

Arguments

Table C.9. Arguments for SetAPSettings

Argument	Direction	relatedStateVariable
NewAPSettings	IN	APSettings

Dependency on State (if any)

Effect on State (if any)

Errors

errorCode	ErrorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.

C.2.4.5 DelAPSettings

This action is used to delete settings or entries on an access point for which the registrar already has established a secure association. The input message is the **DelAPSettings** message in the WCN-NET specification. This action is required for access points.

Arguments

Table C.10. Arguments for DelAPSettings

Argument	Direction	relatedStateVariable
NewAPSettings	In	APSettings

Dependency on State (if any)

Effect on State (if any)

Errors

errorCode	ErrorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.

C.2.4.6 GetSTASettings

This action retrieves the settings from an Ethernet-attached wireless device for which a registrar has already established a secure association. The input message is the **GetSTASettingsInput** message in the WCN-NET specification. The output message is the **GetSTASettings** output message in the WCN-NET specification. It is required for Ethernet-attached wireless stations.

Arguments

Table C.11. Arguments for GetSTASettings

Argument	Direction	relatedStateVariable
Message	In	Message
NewSTASettings	Out	STASettings

Dependency on State (if any)

Effect on State (if any)

Errors

errorCode	ErrorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.

C.2.4.7 SetSTASettings

This action sets settings on an Ethernet-connected wireless device for which a registrar has already established a secure association. The input message is the **SetSTASettings** input message in the WCN-NET specification. It is required for Ethernet-attached wireless stations.

Arguments

Table C.12. Arguments for SetSTASettings

Argument	Direction	relatedStateVariable
NewSTASettings	In	STASettings

Dependency on State (if any)

Effect on State (if any)

Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.

C.2.4.8 DelSTASettings

This action is used to delete settings or entries on an access point for which the registrar already has established a secure association. The input message is the **DelSTASettings** input message in the WCN-NET specification. This action is required for Ethernet-attached wireless stations.

Arguments

Table C.13. Arguments for DelSTASettings

Argument	Direction	relatedStateVariable
NewSTASettings	In	STASettings

Dependency on State (if any)

Effect on State (if any)

Errors

errorCode	ErrorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.

C.2.4.9 PutWLANResponse

This action is used to send messages to a wireless enrollee via the proxy function. The registrar must subscribe to events from the proxy function that forwards enrollee requests and the registrar responds via **PutWLANResponse** to continue the registration protocol exchange.

Arguments

Table C.14 Arguments for PutWLANResponse

Argument	Direction	relatedStateVariable
NewMessage	In	Message
NewWLANEventType	In	WLANEventType
NewWLANEventMAC	In	WLANEventMAC

Dependency on State (if any)

Effect on State (if any)

Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.

C.2.4.10 SetSelectedRegistrar

This action is used to inform the proxy that the registrar has been selected (such as the registrar button has been pushed) and that the proxy indicates to enrollees that a registrar subscribed to its events has been selected. It is required for proxies. The input message is the **SetSelectedRegistrar** message in the WCN-NET specification.

Arguments

Table C.15. Arguments for RebootAP

Argument	Direction	relatedStateVariable
NewMessage	In	Message

Dependency on State (if any)

Effect on State (if any)

Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.

C.2.4.11 RebootAP

This action requests the access point to reboot and may optionally provide a new configuration instead of an existing configuration to be set before reboot. This command requires a previously established secure association between the access point and the registrar. The access point must confirm the validity of the request by checking the hash of the message before performing a reboot. The input message is the **ResetAP** message in the WCN-NET specification.

Arguments

Table C.16 Arguments for RebootAP

Argument	Direction	relatedStateVariable
NewAPSettings	In	APSettings

Dependency on State (if any)

Effect on State (if any)

Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.

C.2.4.12 ResetAP

This action requests the access point to reset its configuration to factory settings. The **APSetting** variable is provided to authenticate the source of the request. This action requires a previously established secure association between the access point and the registrar. The access point must confirm the validity of the request by checking the hash of the message before performing a reboot. The input message is the **ResetAP** message in the WCN-NET specification. This action is required for access points.

Arguments

Table C.17. Arguments for ResetAP

Argument	Direction	relatedStateVariable
NewMessage	In	Message

Dependency on State (if any)

Effect on State (if any)

Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.

C.2.4.13 RebootSTA

This action requests the UPnP station to reboot and may optionally provide a new configuration instead of an existing configuration to be set before reboot. This action requires a previously established secure association between the station and the registrar. The input message is the **ResetAP** message in the WCN-NET specification. The station must confirm the validity of the request by checking the hash of the message before performing a reboot.

Arguments

Table C.18. Arguments for RebootSTA

Argument	Direction	relatedStateVariable
NewSTASettings	In	STASettings

Dependency on State (if any)

Effect on State (if any)

Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.

C.2.4.14 ResetSTA

This action requests the station to reset its configuration to factory settings. The **STASettings** variable is provided to authenticate the source of the request. This action requires a previously established secure association between the station and the registrar. The input message is the **ResetAP** message in the WCN-NET specification. The station must confirm the validity of the request by checking the hash of the message before performing a reboot. This action is required for stations that implement the *WFAWLANConfig* service.

Arguments

Table C.19. Arguments for ResetSTA

Argument	Direction	relatedStateVariable
NewMessage	In	Message

Dependency on State (if any)

Effect on State (if any)

Errors

errorCode	errorDescription	Description
402	Invalid Args	See UPnP Device Architecture section on Control.

C.2.4.15 Nonstandard Actions Implemented by a UPnP Vendor

To facilitate certification, nonstandard actions implemented by UPnP vendors should be included in this service template. The UPnP Device Architecture lists naming requirements for nonstandard actions (see the section on Description).

C.2.4.16 Common Error Codes

Table C.20 lists error codes common to actions for this service type. If an action results in multiple errors, the most specific error should be returned.

Table C.20. Common Error Codes for all actions

ErrorCode	errorDescription	Description
401	Invalid Action	See UPnP Device Architecture section on Control.
402	Invalid Args	See UPnP Device Architecture section on Control.
404	Invalid Var	See UPnP Device Architecture section on Control.
501	Action Failed	See UPnP Device Architecture section on Control.
600-699	TBD	Common action errors. Defined by UPnP Forum Technical Committee.
701-799		Common action errors defined by the UPnP Forum working committees.
800-899	*TBD*	*(Specified by UPnP vendor.)*

C.2.5 Theory of Operation

C.2.5.1 Establishing a Registrar with an Access Point and Access Point Management

When an access point initializes on the network, it advertises the *WFADevice* and the *WFAWLANConfig* service to the UPnP network. A device detects the notification and retrieves the **DeviceInfo** (M1 message in the Registration Protocol) to determine an access point with which it wants to establish itself as a registrar and follows the same WCN-NET process, validating with the access point via a PIN (or password if set) over the **PutMessage** UPnP action. After the secure association is

established between the pair, the registrar may add or remove station configurations or reconfigure the WLAN by using the TLV over UPnP methods defined herein, referencing the secure association via the Session ID (SID) by using the various access point configuration actions.

C.2.5.2 Proxy Function

The proxy function bridges the UPnP network and the UPnP network to forward notification of the existence of a device that wants to receive WLAN configuration. A registrar receives the requests by subscribing to the UPnP event service provided by the proxy function and performs the WCN-NET exchange with the enrollee over UPnP by using the **WLANEvent/WLANEventType/WLANEventMAC** event and the **PutWLANResponse** action.

The proxy function must cache registrar responses to Probe request events and to EAP message exchanges so that the enrollee may get information serially for each registrar that is using the proxy function on a device. The 802.11 mechanisms are described in the WCN-NET specification.

C.2.5.3 Initialization and Configuration of the Ethernet-Connected Wireless Device

When an Ethernet-connected wireless device initializes on the network, it advertises *WFADevice* and the *WFAWLANConfig* service to the UPnP network. A device detects the notification and retrieves **DeviceInfo** to determine that it is a wireless station that needs wireless settings over Ethernet and follows the WCN-NET process, validated the station with a PIN (or password if set) over the UPnP **GetMessage** and **PutMessage** actions. After the secure association is established between the pair, the device can be accessed directly via **GetSTASettings** and **SetSTASettings** at a later time if returned to the Ethernet network or optionally over the wireless network by referencing the Session ID (SID).

C.3 XML Service Description

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
    <specVersion>
        <major>1</major>
        <minor>0</minor>
    </specVersion>
    <actionList>
        <action>
            <name>GetDeviceInfo</name>
            <argumentList>
                <argument>
                    <name>NewDeviceInfo</name>
                    <direction>out</direction>
                  <relatedStateVariable>DeviceInfo</relatedStateVaria
ble>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>PutMessage</name>
            <argumentList>
                <argument>
                    <name>NewInMessage</name>
                    <direction>in</direction>
                  <relatedStateVariable>InMessage</relatedStateVariab
le>
                </argument>
                <argument>
                    <name>NewOutMessage</name>
                    <direction>out</direction>
                  <relatedStateVariable>OutMessage</relatedStateVaria
ble>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>GetAPSettings</name>
            <argumentList>
                <argument>
                    <name>NewMessage</name>
                    <direction>in</direction>
                  <relatedStateVariable>Message</relatedStateVariable
>
                </argument>
                <argument>
                    <name>NewAPSettings</name>
                    <direction>out</direction>
```

```
                    <relatedStateVariable>APSettings</relatedStateVaria
ble>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>SetAPSettings</name>
            <argumentList>
                <argument>
                    <name>APSettings</name>
                    <direction>in</direction>
                    <relatedStateVariable>APSettings</relatedStateVaria
ble>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>DelAPSettings</name>
            <argumentList>
                <argument>
                    <name>NewAPSettings</name>
                    <direction>in</direction>
                    <relatedStateVariable>APSettings</relatedStateVaria
ble>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>GetSTASettings</name>
            <argumentList>
                <argument>
                    <name>NewMessage</name>
                    <direction>in</direction>
                    <relatedStateVariable>Message</relatedStateVariable
>
                </argument>
                <argument>
                    <name>NewSTASettings</name>
                    <direction>out</direction>
                    <relatedStateVariable>STASettings</relatedStateVaria
ble>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>SetSTASettings</name>
            <argumentList>
                <argument>
                    <name>NewSTASettings</name>
                    <direction>out</direction>
```

```
                    <relatedStateVariable>STASettings</relatedStateVaria
ble>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>DelSTASettings</name>
            <argumentList>
                <argument>
                    <name>NewSTASettings</name>
                    <direction>in</direction>
                    <relatedStateVariable>STASettings</relatedStateVaria
ble>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>PutWLANResponse</name>
            <argumentList>
                <argument>
                    <name>NewMessage</name>
                    <direction>in</direction>
                    <relatedStateVariable>Message</relatedStateVariable
>
                </argument>
                <argument>
                    <name>NewWLANEventType</name>
                    <direction>in</direction>
                    <relatedStateVariable>WLANEventType</relatedStateVar
iable>
                </argument>
                <argument>
                    <name>NewWLANEventMAC</name>
                    <direction>in</direction>
                    <relatedStateVariable>WLANEventMAC</relatedStateVari
able>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>SetSelectedRegistrar</name>
            <argumentList>
                <argument>
                    <name>NewMessage</name>
                    <direction>in</direction>
                    <relatedStateVariable>Message</relatedStateVariable
>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>RebootAP</name>
            <argumentList>
                <argument>
                    <name>NewAPSettings</name>
                    <direction>in</direction>
```



```
                    <relatedStateVariable>APSettings</relatedStateVaria
ble>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>ResetAP</name>
            <argumentList>
                <argument>
                    <name>NewMessage</name>
                    <direction>in</direction>
                  <relatedStateVariable>Message</relatedStateVariable
>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>RebootSTA</name>
            <argumentList>
                <argument>
                    <name>NewSTASettings</name>
                    <direction>in</direction>
                  <relatedStateVariable>APSettings</relatedStateVaria
ble>
                </argument>
            </argumentList>
        </action>
        <action>
            <name>ResetSTA</name>
            <argumentList>
                <argument>
                    <name>NewMessage</name>
                    <direction>in</direction>
                  <relatedStateVariable>Message</relatedStateVariable
>
                </argument>
            </argumentList>
        </action>
    </actionList>
    <serviceStateTable>
        <stateVariable sendEvents="no">
            <name>Message</name>
            <dataType>bin.base64</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>InMessage</name>
            <dataType>bin.base64</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>OutMessage</name>
            <dataType>bin.base64</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>DeviceInfo</name>
            <dataType>bin.base64</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>APSettings</name>
            <dataType>bin.base64</dataType>
        </stateVariable>
        <stateVariable sendEvents="yes">
```

```
            <name>APStatus</name>
            <dataType>ui1</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>STASettings</name>
            <dataType>bin.base64</dataType>
        </stateVariable>
        <stateVariable sendEvents="yes">
            <name>STAStatus</name>
            <dataType>ui1</dataType>
        </stateVariable>
        <stateVariable sendEvents="yes">
            <name>WLANEvent</name>
            <dataType>bin.base64</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>WLANEventType</name>
            <dataType>ui1</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>WLANEventMAC</name>
            <dataType>string</dataType>
        </stateVariable>
        <stateVariable sendEvents="no">
            <name>WLANResponse</name>
            <dataType>bin.base64</dataType>
        </stateVariable>
    </serviceStateTable>
</scpd>
```

C.4 Test

No semantic tests have been defined for this service.

MICROSOFT WINDOWS RALLY PROGRAM
LICENSE AGREEMENT

This Microsoft Windows Rally Program License Agreement (the "**Agreement**") is by and between **MICROSOFT CORPORATION**, a corporation organized under the laws of Washington ("**Microsoft**"), and the individual or entity identified and signing below ("**You**" or "**Licensee**"). ***If You want a license from Microsoft to access, view or implement one or more Licensed Technologies (as defined below), You must (1) complete the designated information in the box below, and (2) sign and return this Agreement AS IS to Microsoft at the address shown in Section 8.3***. This is an offer to be accepted only on the terms set forth in this Agreement. If any other changes are made to this Agreement, the offer is revoked. This Agreement, completed and fully executed by You, will become effective on the date it is received by Microsoft pursuant to Section 8.3 (the "**Effective Date**").

Licensee full legal name:
Type of legal entity (corporation, partnership, sole proprietorship, individual or other):
State/Province organized:
Street address:
City, State & Country:
Email contact for notices:

Recitals

Microsoft has developed certain "Windows Rally" technologies, including communications protocols, that are intended to facilitate certain tasks performed by or in conjunction with network connected devices. Licensee desires a license from Microsoft to access, view or implement the Licensed Technology(ies) for which the applicable box(es) are checked on **Exhibit A**, and to use the corresponding Technical Documentation (as defined below) for the purpose of such implementation(s).

1. Definitions

1.1 "Affiliate" means, with respect to any legal entity, any other such entity directly or indirectly Controlling, Controlled by, or under common Control with such entity. "Control" means the possession, directly or indirectly, of the power to direct or cause the direction of the management and policies of a legally recognizable entity, whether through the ownership of voting shares or other voting interests, by contract, or otherwise.

1.2 "Authorized Purpose" means, with respect to each Licensed Technology, the function(s) corresponding to such Licensed Technology as set forth in **Exhibit A.**

1.3 "Licensed Implementation(s)" means only those specific portion(s) of Licensee's products that (a) implement Licensed Technology(ies) solely to carry out the corresponding Authorized Purpose(s), and (b) are compliant with all Required Portions of the relevant Technical Documentation.

1.4 "Licensed Product" means a product, branded with a trademark owned or controlled by Licensee, that includes Licensed Implementation(s) either alone or with other components.

1.5 "Licensed Technology(ies)" means the version(s) of the software communications protocol(s) and related technology(ies) which Licensee indicates it has selected from the list on **Exhibit A**. "Licensed Technology(ies)" also includes any version(s) of such protocol(s) or technology(ies) and any new protocol(s) or technology(ies) that may be added to this Agreement by mutual written agreement between Licensee and Microsoft in the future.

1.6 "Necessary Claims" means claims of an unexpired patent or patent application that (a) are owned or controlled by Microsoft or a Microsoft Affiliate; and (b) are necessarily infringed by implementing the Required Portions of the Technical Documentation, wherein a claim is necessarily infringed only when it is not possible to avoid infringing it because there is no non-infringing alternative for implementing the Required Portion of the relevant Technical Documentation. "Necessary Claims" do not include any claims: (i) other than those set forth above even if contained in the same patent as Necessary Claims; (ii) that, if licensed, would require a payment of royalties or other fee(s) by a party to unaffiliated third parties; (iii) to

portion thereof that implements the relevant Licensed Technology(ies); or (iv) to any implementation of other technical documentation, specifications or technologies that are merely referred to in the body of the Technical Documentation.

1.7 "Required Portions" means all portions of Technical Documentation with the exception of any portions that are expressly identified by Microsoft as being "optional."

1.8 "Technical Documentation" means Microsoft's proprietary technical documentation for the Licensed Technology(ies).

2. Windows Rally Program Administration; Program Additions and Updates

2.1 Delivery. Microsoft has made the Technical Documentation available to Windows Rally program licensees at www.microsoft.com/rally. The Technical Documentation shall be deemed accepted by Licensee upon its download.

2.2 Program Additions and Updates. Microsoft may in its discretion elect to include additional software communications protocols or other technologies in the Windows Rally program in the future by publishing technical documentation for them at www.microsoft.com/rally and making available amended terms of this Agreement at such website in order to make such protocols or other technologies available under this Agreement. Except in the event Microsoft makes such additional protocols or other technologies available and Licensee accepts applicable terms governing their inclusion under this Agreement, no other protocols, technologies, technical documentation, or Microsoft enhancements or updates to the Licensed Technology(ies) and/or Technical Documentation are licensed under this Agreement.

2.3 No Technical Support. This Agreement does not include technical support by Microsoft to Licensee, its channel entities, or end users. Licensee is solely responsible for all such support and shall advise channel entities and end users accordingly.

3. Licenses

3.1 Copyright License. Microsoft hereby grants Licensee a non-exclusive, royalty-free, non-sublicensable, nontransferable, personal, worldwide license to make a reasonable number of complete copies of the Technical Documentation for use solely in developing and testing such Licensed Implementation(s). Licensee shall retain in all copies of, and shall comply with, any copyright-related notices that are included in such Technical Documentation.

3.2 Patent License. Microsoft hereby grants Licensee a nonexclusive, royalty-free, non-sublicensable, nontransferable, personal, worldwide license under the Necessary Claims to use the Technical Documentation for the Licensed Technology(ies)s to:

(a) make, use, import, offer to sell, sell and distribute directly or indirectly to end users, object code versions of Licensed Implementations only as incorporated into Licensed Products and solely for the purpose of conforming with the corresponding Technical Documentation in order for Licensed Implementations to carry out the Authorized Purpose corresponding to such Licensed Technology(ies), and

(b) to distribute or otherwise disclose source code copies of the Licensed Implementation(s) licensed in Section 3.2(a) only if Licensee (i) prominently displays the following notice in all copies of such source code, and (ii) distributes or discloses the source code only under a license agreement that includes the following notice as a term of such license agreement and does not include any other terms that are inconsistent with, or would prohibit, the following notice:

"This source code incorporates intellectual property owned by Microsoft Corporation and cannot be made, used, sold, offered for sale, imported or redistributed without a license from Microsoft Corporation. Our provision of this source code does not include any licenses or any other rights to you under any Microsoft intellectual property. If you would like a license from Microsoft, you need to contact Microsoft directly (send mail to rally@microsoft.com)."

3.3 License Clarifications. The licenses granted to Licensee in this Agreement do not include any right to (i) modify the Technical Documentation, change any of the packet types or content types described in the Technical Documentation, or extend any such packet types or content types except as described in the

Necessary Claims in any software other than a Licensed Implementation or to support any communications between computing devices and/or computing functions other than as expressly provided in the definition of the Authorized Purpose corresponding to the applicable Licensed Technology(ies). For the avoidance of doubt, regardless of whether a Licensed Implementation performs or is capable of performing functions other than the Authorized Purpose, the licenses set forth in Section 3.2 apply only to the performance of the Authorized Purpose corresponding to such Licensed Implementation.

3.4 Licensee Covenant Not to Sue. In consideration of the terms of this Agreement, Licensee covenants on behalf of itself and its Affiliates that it will not sue Microsoft or any Microsoft Affiliate for infringement of any claims of patents reading on any Technical Documentation ("**Licensee Patents**") on account of any manufacture, use, sale, offer for sale, importation or other disposition or promotion, worldwide, of any Microsoft product, technology, service or portion thereof. If Licensee assigns or transfers any Licensee Patents, applications therefor, or rights to enforce the same, Licensee shall require as a condition of any such assignment that the assignee agrees to be bound by the provisions of this Section 3.4 with respect to Licensee Patents. Any purported assignment or transfer of rights in derogation of the foregoing requirements shall be null and void.

3.5 No Warranties. Licensee shall make no representation, or any express or implied warranty to third parties (including, without limitation, to any end users) on behalf of Microsoft.

3.6 Excluded Licenses. Nothing in this Agreement authorizes Licensee to subject the Licensed Technology(ies), Technical Documentation or any Microsoft intellectual property right in any Licensed Implementation to the terms of any license that requires, as a condition of use, modification or distribution of technology subject to such license, that such technology or other technology combined or distributed with such technology (a) be disclosed or distributed in source code form; (b) be licensed for the purpose of making derivative works; or (c) be re-distributable at no charge.

3.7 Reservation of Rights. All rights not expressly granted in this Agreement are reserved. No additional rights whatsoever (including, without limitation, any implied licenses) are granted by implication, estoppel or otherwise. Without limiting the generality of the foregoing, this Agreement does not grant Licensee any license or other right to use or display any name, trade name, logo, trademark or other identifier of Microsoft (provided, however, that this Agreement also does not restrict any right that Licensee may have under applicable laws to make accurate, descriptive and nominative references to Microsoft or its products in any press release, public announcement or other disclosure, including pursuant to Section 7).

4. Term and Termination

4.1 Term. The term of this Agreement shall commence as of the Effective Date and continue unless and until terminated in accordance with the provisions of this Agreement.

4.2 Termination. (a) Licensee may terminate this Agreement at any time upon written notice to Microsoft. Microsoft may terminate this Agreement (i) immediately upon written notice at any time, if Licensee is in material breach of Section 3.2(b); or (ii) if Licensee otherwise materially breaches this Agreement and fails to cure the breach within thirty (30) days after Licensee receives notice of the breach from Microsoft. (b) Upon termination, Licensee's licenses under this Agreement shall end and Licensee shall cease all use of the Technical Documentation (including but not limited to all production and all distribution of Licensed Implementations and Licensed Products).

4.3 Survival. Sections 1, 2.2, 3.4, 3.6, 3.7, 4.2, 5.2, 6, 8 and this Section 4.3 shall survive any termination of this Agreement. Licenses granted prior to the termination of this Agreement by Licensee to end users for Licensed Implementations in accordance with the terms of this Agreement shall survive any termination of this Agreement.

5. Representations and Disclaimers of Warranty

5.1 Licensee represents and warrants that the person signing this Agreement on Licensee's behalf has all necessary power and authority to do so, and that upon such signature this Agreement is a binding obligation on Licensee.

5.2 DISCLAIMERS. THE TECHNICAL DOCUMENTATION, LICENSED TECHNOLOGY(IES) AND ALL INTELLECTUAL PROPERTY MADE AVAILABLE AND/OR LICENSED BY MICROSOFT UNDER OR IN CONNECTION WITH THIS AGREEMENT ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. MICROSOFT DISCLAIMS ALL WARRANTIES, DUTIES AND CONDITIONS, EITHER EXPRESS, IMPLIED OR STATUTORY WITH RESPECT TO SUCH TECHNICAL DOCUMENTATION, LICENSED TECHNOLOGY(IES) AND INTELLECTUAL PROPERTY, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NON-INFRINGEMENT.

6. Limitation of Liability

MICROSOFT SHALL NOT BE LIABLE FOR ANY DAMAGES ARISING FROM OR OTHERWISE RELATED TO THIS AGREEMENT, INCLUDING INDIRECT, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR SPECIAL DAMAGES, EVEN IF MICROSOFT HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES IN ADVANCE.

7. Marketing

7.1 Press Release.

(a) Promptly following the Effective Date, Licensee will issue a press release in which Licensee announces that it has entered into a Microsoft Windows Rally Program License Agreement and describes Windows Rally technology as being intended to provide effortless, secure and reliable connectivity between devices and PCs, support for new connectivity scenarios, and richer experiences for customers.

(b) The parties may cooperate with each other on press releases and similar communications regarding the non-confidential subject matter of this Agreement. The content and specific timing of each such press release (including the press release to be issued pursuant to Section 7.1(a)) or similar public communications must be agreed upon in writing by both parties.

7.2 Microsoft Marketing Materials. Notwithstanding Section 7.1(b), Licensee consents to Microsoft's listing Licensee as a party to its Microsoft Windows Rally Program License Agreement on the Microsoft Windows Rally website, as well as in marketing materials generated by or for Microsoft.

8. Miscellaneous

8.1 No Partnership, Joint Venture or Franchise. Neither this Agreement, nor any terms and conditions contained herein, shall be construed as creating a partnership, joint venture or agency relationship or as granting a franchise as defined in the Washington Franchise Investment Protection Act, RCW 19.100, as amended, or 16 CFR Section 436.2(a), or any similar laws in other jurisdictions.

8.2 Export Regulations. Licensee acknowledges that the Technical Documentation and Licensed Implementations may be subject to U.S. export jurisdiction and other applicable national or international laws. Licensee agrees to comply with all applicable international and national laws that apply to the Technical Documentation and Licensed Implementations, including the U.S. Export Administration Regulations, as well as end-user, end-use and destination restrictions issued by U.S. and other governments, and privacy laws. See http://www.microsoft.com/exporting/.

8.3 Executed Agreements and Effectiveness; Notices. To be effective, executed Agreement(s) must be sent by messenger, traceable express mail or prepaid certified mail, return receipt requested, addressed to Microsoft as follows:

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399
Attention: Windows Rally Program Manager (Scott Manchester or successor)
Copy to: Law & Corporate Affairs

Except for the foregoing, all notices in connection with this Agreement shall be deemed given as of the day they are received either by messenger, delivery service, or in the United States of America mails, postage prepaid, certified or registered, return receipt requested, and addressed either to Licensee as stated in the box on the first page of this Agreement or to Microsoft as stated above, or to such other address as a party may designate pursuant to this notice provision.

8.4 Governing Law; Jurisdiction; Attorneys' Fees. This Agreement shall be construed and controlled by the laws of the State of Washington, and Licensee consents to exclusive jurisdiction and venue in the federal courts sitting in King County, Washington, unless no federal subject matter jurisdiction exists, in which case Licensee consents to exclusive jurisdiction and venue in the Superior Court of King County, Washington. Licensee waives all defenses of lack of personal jurisdiction and forum non conveniens. Process may be served on either party in the manner authorized by applicable law or court rule.

8.5 Assignment. Licensee shall not transfer or assign this Agreement, or any rights or obligations hereunder, whether by operation of contract, law or otherwise, except with the express written consent of Microsoft, and any attempted assignment by Licensee in violation of this Section is void. For purposes of this Agreement, an "assignment" by Licensee is deemed to include, without limitation, each of the following: (a) a change in beneficial ownership of Licensee of greater than twenty percent (20%) (whether in a single transaction or series of transactions) if Licensee is a partnership, trust, limited liability company or other like entity; (b) a merger of Licensee with another party, whether or not Licensee is the surviving entity; (c) the acquisition of more than twenty percent (20%) of any class of Licensee's voting stock (or any class of non-voting security convertible into voting stock) by another party (whether in a single transaction or series of transactions); and (d) the sale or other transfer of more than fifty percent (50%) of Licensee's assets (whether in a single transaction or series of transactions). In the event of such assignment or attempted assignment by Licensee, Microsoft may, but is not obligated to, immediately terminate this Agreement and any licenses or rights grants hereunder.

8.6 Construction. If for any reason a court of competent jurisdiction finds any provision of this agreement, or portion thereof, to be unenforceable (other than Section 3.2(b)), such provision and the rest of the Agreement will be enforced to the maximum extent permissible so as to effect the intent of the parties, and the Agreement will continue in full force and effect. In the event that a court of competent jurisdiction finds that Section 3.2(b) is unenforceable, this entire Agreement and any licenses granted hereunder shall be rendered null and void. Failure by a party to enforce any provision of this Agreement will not be deemed a waiver of future enforcement of that or any other provision.

8.7 No Requirement to Implement. Nothing in this Agreement shall be construed as requiring Licensee to use or implement Licensed Technology(ies), or limit either party from competing in any way without infringing the intellectual property rights of the other party, including by engaging in activities, independently or with others, that may be deemed competitive with Licensed Technology(ies), Licensed Implementation(s) or Licensed Product(s).

8.8 Entire Agreement. This Agreement constitutes the entire agreement between the parties with respect to its subject matter, and merges all prior and contemporaneous communications. It shall not be modified except by a written agreement dated subsequent to the date of this Agreement and signed on behalf of Licensee and Microsoft by their respective duly authorized representatives.

By signing below Licensee agrees to the foregoing and represents that Licensee has not modified this Agreement in any way.

By (signature):	Licensee Name:
Name (printed):	Dated:
Title:	

EXHIBIT A
LICENSED TECHNOLOGY(IES)

The Licensed Technology(ies) that Licensee has chosen to license under this Agreement are indicated by check(s) in the box(es) on the left. Each Licensed Implementation is licensed to perform only the corresponding Authorized Purpose and is subject to the further terms of this Agreement.

	Technology	Authorized Purpose
☐	Link Layer Topology Discovery	Licensed Implementations of this Licensed Technology may use such Licensed Technology only for the following purpose: to request or respond to requests (in either case in communications with other devices that include a Licensed Implementation) for information describing (i) the device on which the Licensed Implementation is running, (ii) information about other devices connected to or which have wireless access to the same network as the device on which the Licensed Implementation is running, and (iii) characteristics about the physical media to which the device in (i) is connected.
☐	PnP-X	Licensed Implementations of this Licensed Technology may use such Licensed Technology only for the following purpose: to implement the methodology described in the Technical Documentation for how specific communications protocols described in (but not licensed under) such Technical Documentation can be extended and employed to provide a plug and play experience.
☐	WCN	Licensed Implementations of this Licensed Technology may use such Licensed Technology only for the following purpose: to permit the transfer of Windows Connect Now Settings to a Licensed Product from a computing device and then from such Licensed Product to other devices.
☐	DPWS	Licensed Implementations of this Licensed Technology may use such Licensed Technology only for the following purpose: to perform the functions of a Compliant Device or a Compliant Control Point, as defined below. • A Compliant Device means a Licensed Product that includes a Licensed Implementation which implements the mandatory aspects of at least one DCP (as defined below) for the sole purpose of responding to requests from a Compliant Control Point. • A Compliant Control Point means a Licensed Product that includes a Licensed Implementation the sole function of which is to send requests to a Compliant Device. • A DCP means a device control protocol, including a device-specific schema described using the web-services description language, which device control protocol enables interaction between a device associated with one device class and devices associated with the same or different device classes in a networked environment.