RFC 209 - NetworkInterfaceInformationService

Draft

28 Pages

Abstract

This document defines the Java API that provides the information of network interfaces in an OSGi environment. The bundles can get not only information of network interfaces but notification when the configuration of network interfaces to be changed to use this API.

0 Document Information

0.1 License

DISTRIBUTION AND FEEDBACK LICENSE, Version 2.0

0.2 Trademarks

0.3 Feedback

This document can be downloaded from the OSGi Alliance design repository at https://github.com/osgi/design The public can provide feedback about this document by opening a bug at https://www.osgi.org/bugzilla/.

0.4 Table of Contents

0.5 Terminology and Document Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in 10.1.

`Source code is shown in this typeface.`

0.6 Revision History

The last named individual in this history is currently responsible for this document.

Revision	Date	Comments
Initial	Nov 18, 2013	Initial version Shigekuni Kondo, NTT Corporation, kondo.shigekuni@lab.ntt.co.jp

1 Introduction

Java standard APIs (i.e. java.net.NetworkInterface, java.net.InetAddress) provide functions that allow IP network interface information, such as the IP address and MAC address to be obtained.

However, the bundle that wants to get network interface information has to monitor whether the information has changed or not for a certain period of time. Changes in network interface can be pushed to the bundles concerned, the need for polling by bundles can be eliminated.

In addition, some information cannot be obtained via Java standard APIs.

This RFC

defines the Java API that provides the information of network interfaces in an OSGi environment. The bundles can get not only information of network interfaces but notification when the configuration of network interfaces to use this API.

2 Application Domain

There are many bundles that use the IP network to communicate with other networked devices. In paticular, since a Residential Gateway (RGW) may have a number of network interfaces, each bundle running on the RGW needs to obtain an IP address and confirm whether the network interface associated with the allocated IP address suits the bundle's requirements or not.

For example, a protocol adapter needs the IP address of a network interface on the wide area network side to communicate with an external server. UPnP device service bundle needs the IP address that can be used to communicate with devices in a local area network.

These bundles can acquire information about the network interface via the following Java standard APIs.

- java.net.NetworkInterface
- *java.net.InetAddress*

3 Problem Description

Many application bundles on the RGW provide services on IP networks. For example, a protocol adapter for DMT Admin Service, a http server established by HTTP Service bundle and UPnP device service bundle use IP networks. In those cases, the bundles need to get information about the network interface on the RGW such as IP address, MAC address, network interface name, and so on.

The information about the network interface can be obtained by using Java standard APIs which are java.net.NetworkInterface and java.net.InetAddress. However, these APIs fail to provide the features needed by the bundles when they use the IP network in the following situations:

[Problem 1] There is no feature that sends a notification when information of the network interface (i.e. IP address) changes during runtime, e.g. the connection status or the assigned IP address.

[Problem 2] There is no feature that can acquire the subnet mask of the network interface.

[Problem 3] Operating System specific bundles must be prepared because some information about network interface depends on the Operating System.

If these functions were available, it would be very useful for bundles that need to use the IP network. However, a standard API does not exist at this time, so it must be prepared for each environment.

3.1 Use Cases

Use case 1

The TR-069 protocol adapter bundle on a RGW needs to communicate with an Auto Configuration Server (ACS). The ACS needs to know the public IP address of the Residential Gateway to send a UDP packet to the protocol adapter bundle for a connection request. In this case, the bundle has to provide the IP address to the ACS when the bundle is started or the IP address has changed.

Use case 2

When an HTTP Service bundle is available, at least one HTTP server is expected to run. When the HTTP server needs to be assigned to a specific network interface, the HTTP Service bundle has to know the information of the network interface. In addition, the HTTP Service bundle needs to know when the IP address of the network interface being managed changes.

Use case 3

The UPnP Device Service bundle needs to create the DatagramSocket for receiving and sending M-search messages. In the case of devices such as Residential Gateway, which has multi network interfaces, the UPnP bundle has to create a DatagramSocket that is bound to an appropriate local IP address. Therefore, the UPnP bundle needs to know the current IP address of the network interface and the replacement IP address.

Use case 4

An application bundle wants to obtain the subnet mask of the IP address to cover the situation in which the bundle needs to execute the Wake-up-On-LAN process.

Use case 5

An application wants to obtain information about available network services, such as available DNS Server, Log Server, NTP Server, or network characteristics, such as domain names, ARP cache timeouts, broadcast address, etc. For this, the local DHCP server can be queried to get those information.

Use case 6

A device running an OSGi framework in an mixed IPv4/IPv6 environment needs to get specific information about the network interface(s) in order to provide, for example, different services for the IPv4 and IPv6 environments.

4 Requirements

[[REQ_1] The solution MUST provide means to send notifications to interested bundles whenever the information of network interface has changed.(i.e. The bundle is notified the information of IP address change from Network Interface Information Service implemented bundle)

[REQ_2] The solution MUST provide an API that can obtain information from a multiple network interfaces. Each network interface can provide information about multiple addresses. (An application bundle needs to know whether the network interface is a LAN interface or a WAN interface.).

[REQ_3] The solution MUST provide a mechanism that can provide the network interface information needed regardless of the Operating System type.

[REQ_4] The solution MUST provide the means of configuring network interface type. It will be defined for each environment (i.e. “LAN”, “WAN” that is bound to each logical interface) .

[REQ_5] The solution MUST provide an API that can obtain the subnet mask of each IP address.

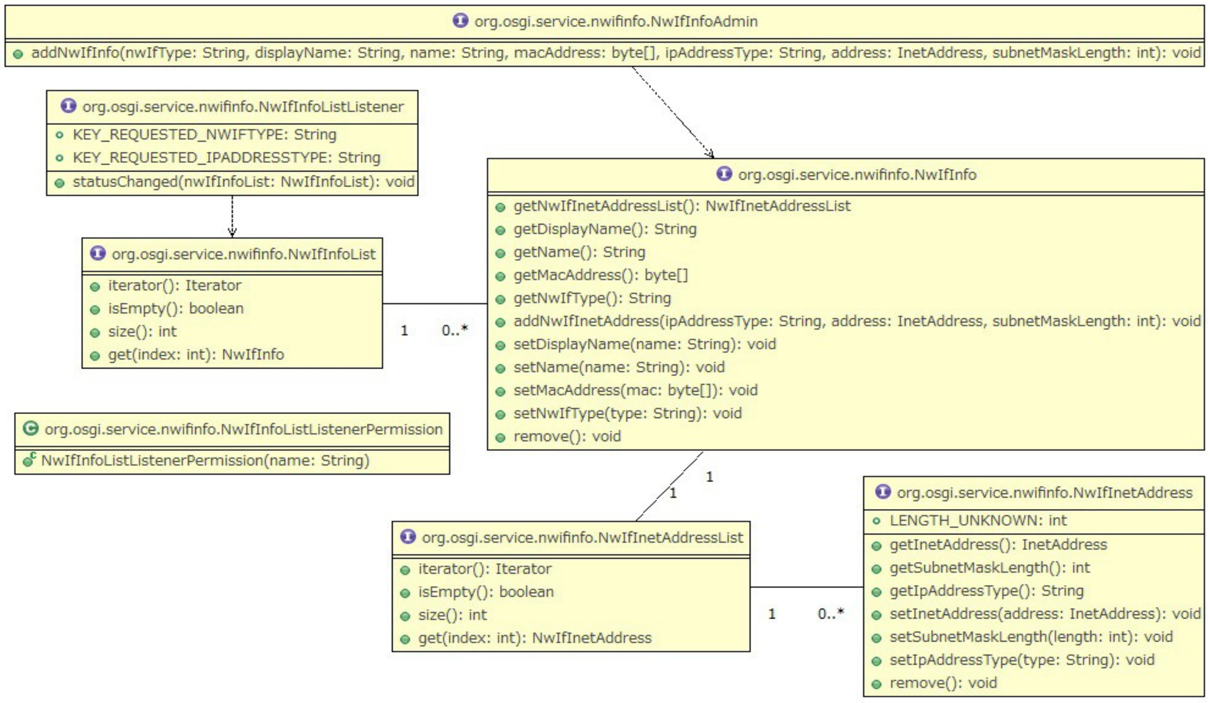
[REQ_6] The solution MUST support both IPv4 and IPv6 environments (mixed or separately) and the corresponding characteristics, for example IPv4 and IPv6 addresses, multi-prefixes, multicast etc. .

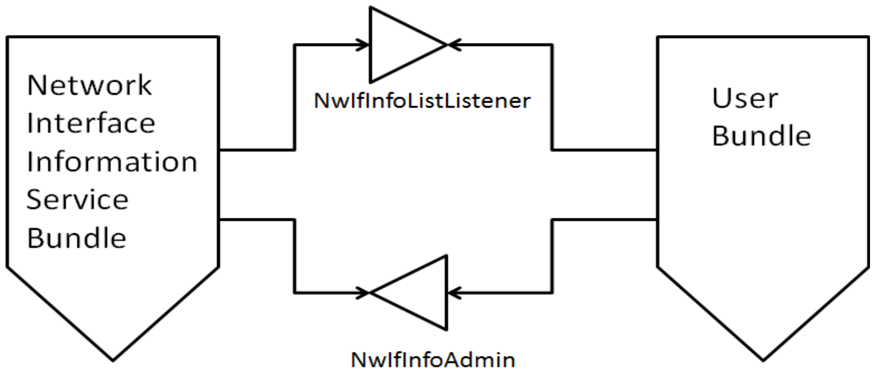
[REQ_7] The solution SHOULD support the retrieval of MAC addresses for network interfaces.

[REQ_8] The solution MAY provide an API that allows alteration of network interface configurations.

[REQ_9] The solution MAY provide an API that can obtain the capability of network interface. (e.g. the physical type of network interface, list of BOOTP/DHCP command options, DNS server address, Default Gateway address, etc.)

5 Technical Solution


org.osgi.service.nwifinfo.NwIfInfoAdmin
addNwIfInfo(nwIfType: String, displayName: String, name: String, macAddress: byte[], ipAddressType: String, address: InetAddress, subnetMaskLength: int): void
org.osgi.service.nwifinfo.NwIfInfoListListener
KEY_REQUESTED_NWIFTYPE: String
KEY_REQUESTED_IPADDRESSTYPE: String
statusChanged(nwIfInfoList: NwIfInfoList): void
org.osgi.service.nwifinfo.NwIfInfoList
iterator(): Iterator
isEmpty(): boolean
size(): int
get(index: int): NwIfInfo
1
0..*
org.osgi.service.nwifinfo.NwIfInfo
getNwIfInetAddressList(): NwIfInetAddressList
getDisplayName(): String
getName(): String
getMacAddress(): byte[]
getNwIfType(): String
addNwIfInetAddress(ipAddressType: String, address: InetAddress, subnetMaskLength: int): void
setDisplayName(name: String): void
setName(name: String): void
setMacAddress(mac: byte[]): void
setNwIfType(type: String): void
remove(): void
org.osgi.service.nwifinfo.NwIfInfoListListenerPermission
NwIfInfoListListenerPermission(name: String)
1
1
org.osgi.service.nwifinfo.NwIfInetAddressList
iterator(): Iterator
isEmpty(): boolean
size(): int
get(index: int): NwIfInetAddress
1
0..*
org.osgi.service.nwifinfo.NwIfInetAddress
LENGTH_UNKNOWN: int
getInetAddress(): InetAddress
getSubnetMaskLength(): int
getIpAddressType(): String
setInetAddress(address: InetAddress): void
setSubnetMaskLength(length: int): void
setIpAddressType(type: String): void
remove(): void



5.1 NwIfInfo

This interface represents network interface information. Same units as java.net.NetworkInterface.

In addition to that, this interface provides the functionality to change network interface configuration.

5.2 NwIfInfoAdmin

This interface provides the functionality to create new logical network interface. IP address and desplay name etc can be also configured via NwIfInfo interface.

5.3 NwIfInfoList

This interface represents a list of network IF information. That can be obtained NwIfInfo list at that time. with only some of the functions of the java.util.List.

5.4 NwIfInfoListListener

The listener interface to get notification of the list including network interface information that changes dynamically in the device (i.e. Residential Gateway, PC). The bundle that wants to get the status of the network interface (i.e. IP address, MAC address) will implement this interface and register it to the framework as OSGi service . The information corresponding to "type of network interface" and "type of IP address" are specified in service property is notified.

5.5 NwIfInetAddress

This interface represents an IP address information.

5.6 NwInetAddressList

This Interface represents a list of IP addresses for network IF.

This is NwIfInetAddress list at the time the NwIfInfo service implementation bundle detects a state change. With only some of the functions of the java.util.List.

5.7 NwIfInfoListListenerPermission

This class represents access authority of the bundle which registers NwIfInfoListListener service.

The NwIfInfoListListener service must be registered with service property including requested type of network interface and requested type of IP address. In case that the bundle has NwIfInfoListListenerPermission which is set corresponding type of network interface and type of IP address, the NwIfInfoListListener service can be called back with the required network interface information.

6 Data Transfer Objects

T.B.D.

7 Javadoc

OSGi Javadoc

13/11/18 14:00

Package Summary		Page
org.osgi.service.nwifinfo		12

Package org.osgi.service.nwifinfo

Interface Summary		Page
NwIfInetAddress	This interface represents an IP address information.	*13*
NwIfInetAddressList	This Interface represents a list of IP addresses for network IF.	*15*
NwIfInfo	This interface represents network interface information.	*17*
NwIfInfoAdmin		*20*
NwIfInfoList	This interface represents a list of network IF information.	*21*
NwIfInfoListListener	The listener interface to get notification of the list including network interface information that changes dynamically in the device (i.e.	*23*

Class Summary		Page
NwIfInfoListListenerPermission	This class represents access authority of the bundle which registers NwIfInfoListListener service.	*26*

Interface NwIfInetAddress

org.osgi.service.nwifinfo

```
public interface NwIfInetAddress
```

This interface represents an IP address information.

Field Summary		Page
int	**LENGTH_UNKNOWN** Constant indicating that the IPv6 prefix length or IPv4 subnet mask length is unknown.	13

Method Summary		Page
InetAddress	**getInetAddress**() Returns the InetAddress object corresponding to this object.	13
String	**getIpAddressType**() Returns a string that represents the type of this IP address such as "IPV4_GLOBAL".	14
int	**getSubnetMaskLength**() Returns the length of the IPv6 prefix length or subnet mask of the IPv4 address.	14
void	**remove**()	14
void	**setInetAddress**(InetAddress address)	14
void	**setIpAddressType**(String type)	14
void	**setSubnetMaskLength**(int length)	14

Field Detail

LENGTH_UNKNOWN

```
public static final int LENGTH_UNKNOWN = -1
```

Constant indicating that the IPv6 prefix length or IPv4 subnet mask length is unknown.

Method Detail

getInetAddress

```
InetAddress getInetAddress()
```

Returns the InetAddress object corresponding to this object.
Can use InetAddress to obtain such as IP address.

[Note]

This return value InetAddress.getAddress() MUST return same IP address when NwIFInfo Bundle detects a state change.

Returns:
InetAddress

---

getSubnetMaskLength

```
int getSubnetMaskLength()
```

Returns the length of the IPv6 prefix length or subnet mask of the IPv4 address. Returns `LENGTH_UNKNOWN` if length is unknown.

Returns:
IPv4 subnet mask length or IPv6 prefix length

---

getIpAddressType

```
String getIpAddressType()
```

Returns a string that represents the type of this IP address such as "IPV4_GLOBAL". As this interface definition, the meaning of the string returned by this method is not defined except that it returns null if unknown.

Returns:
IP address Type

---

setInetAddress

```
void setInetAddress(InetAddress address)
```

---

setSubnetMaskLength

```
void setSubnetMaskLength(int length)
```

---

setIpAddressType

```
void setIpAddressType(String type)
```

---

remove

```
void remove()
```

Interface NwIfInetAddressList

org.osgi.service.nwifinfo

`public interface` **`NwIfInetAddressList`**

This Interface represents a list of IP addresses for network IF.
This is NwIfInetAddress list at the time the NwIfInfo service implementation bundle detects a state change. With only some of the functions of the java.util.List (object implements this interface is the Immutable).

Method Summary		*Page*
`NwIfInetAddress`	**`get`**`(int index)` Returns the element(`NwIfInetAddress`) at the specified position in this list.	*16*
`boolean`	**`isEmpty`**`()` Returns true if this list contains no elements.	*15*
`Iterator`	**`iterator`**`()` Returns an iterator stored `NwIfInetAddress`.	*15*
`int`	**`size`**`()` Returns the number of elements in this list Corresponds List#size().	*15*

Method Detail

iterator

`Iterator` **`iterator`**`()`

Returns an iterator stored `NwIfInetAddress`.
Corresponds List#iterator().

Returns:
an iterator stored `NwIfInetAddress`

isEmpty

`boolean` **`isEmpty`**`()`

Returns true if this list contains no elements.
Corresponds List#isEmpty().

Returns:
true if this list contains no elements

size

`int` **`size`**`()`

Returns the number of elements in this list
Corresponds List#size().

Returns:
the number of elements in this list

---

get

`NwIfInetAddress` **`get`**`(int index)`

Returns the element(`NwIfInetAddress`) at the specified position in this list. Corresponds List#get(int).

Parameters:
`index` - index of the element to return

Returns:
the element(`NwIfInetAddress`) at the specified position in this list

Interface NwIfInfo

org.osgi.service.nwifinfo

```
public interface NwIfInfo
```

This interface represents network interface information.
Same units as java.net.NetworkInterface.

Method Summary

		Page
void	**addNwIfInetAddress**(String ipAddressType, InetAddress address, int subnetMaskLength)	18
String	**getDisplayName**() Returns the display name of this network IF.	17
byte[]	**getMacAddress**() Returns a byte array containing the MAC address of this network IF.	18
String	**getName**() Get the display name of this network interface such as "eth0".	18
NwIfInetAddressList	**getNwIfInetAddressList**() Returns a list of IP address information of the network IF as NwIfInetAddressList.	17
String	**getNwIfType**() Returns a string that represents the type of this network IF.	18
void	**remove**()	19
void	**setDisplayName**(String name)	18
void	**setMacAddress**(byte[] mac)	18
void	**setName**(String name)	18
void	**setNwIfType**(String type)	19

Method Detail

getNwIfInetAddressList

```
NwIfInetAddressList getNwIfInetAddressList()
```

Returns a list of IP address information of the network IF as NwIfInetAddressList. Corresponds NetworkInterface.getInetAddresses(). Returns NwIfInetAddressList that has zero element if there is no IP address.

Returns:
a list of IP address information of the network IF

getDisplayName

```
String getDisplayName()
```

Returns the display name of this network IF.
Corresponds NetworkInterface.getDisplayName().

Returns:
the display name of this network IF

---

getName

```
String getName()
```

Get the display name of this network interface such as "eth0".
Corresponds NetworkInterface.getName().

Returns:
the display name of this network interface

---

getMacAddress

```
byte[] getMacAddress()
```

Returns a byte array containing the MAC address of this network IF.
Returns in the same format as java.net.NetworkInterface#getHardwareAddress() of Java6.
Returns null if the information can not be acquired.

Returns:
a byte array containing the MAC address

---

getNwIfType

```
String getNwIfType()
```

Returns a string that represents the type of this network IF.
As this interface definition, the meaning of the string returned by this method is not defined except that it returns null if unknown.

Returns:
network IF type

---

addNwIfInetAddress

```
void addNwIfInetAddress(String ipAddressType,
                        InetAddress address,
                        int subnetMaskLength)
```

---

setDisplayName

```
void setDisplayName(String name)
```

---

setName

```
void setName(String name)
```

---

setMacAddress

```
void setMacAddress(byte[] mac)
```

setNwIfType

```
void setNwIfType(String type)
```

remove

```
void remove()
```

Interface NwIfInfoAdmin

org.osgi.service.nwifinfo

```
public interface NwIfInfoAdmin
```

Method Summary		Page
void	**addNwIfInfo**(String nwIfType, String displayName, String name, byte[] macAddress, String ipAddressType, InetAddress address, int subnetMaskLength)	20

Method Detail

addNwIfInfo

```
void addNwIfInfo(String nwIfType,
                 String displayName,
                 String name,
                 byte[] macAddress,
                 String ipAddressType,
                 InetAddress address,
                 int subnetMaskLength)
```

Interface NwIfInfoList

org.osgi.service.nwifinfo

```
public interface NwIfInfoList
```

This interface represents a list of network IF information.
That can be obtained NwIfInfo list at that time. with only some of the functions of the java.util.List (object implements this interface is the Immutable).

Method Summary		***Page***
`NwIfInfo`	`get(int index)` Returns the element(`NwIfInfo`) at the specified position in this list.	*22*
`boolean`	`isEmpty()` Returns true if this list contains no elements.	*21*
`Iterator`	`iterator()` Returns an iterator stored `NwIfInfo`.	*21*
`int`	`size()` Returns the number of elements in this list.	*21*

Method Detail

iterator

```
Iterator iterator()
```

Returns an iterator stored `NwIfInfo`.
Corresponds List#iterator().

Returns:
an iterator stored `NwIfInfo`

isEmpty

```
boolean isEmpty()
```

Returns true if this list contains no elements.
Corresponds List#isEmpty().

Returns:
true if this list contains no elements

size

```
int size()
```

Returns the number of elements in this list.
Corresponds List#size().

Returns:
the number of elements in this list

---

get

`NwIfInfo get(int index)`

Returns the element(`NwIfInfo`) at the specified position in this list. Corresponds List#get(int).

Parameters:
`index` - index of the element to return

Returns:
the element(`NwIfInfo`) at the specified position in this list

Interface NwIfInfoListListener

org.osgi.service.nwifinfo

`public interface` **`NwIfInfoListListener`**

The listener interface to get notification of the list including network interface information that changes dynamically in the device (i.e. Residential Gateway, PC).
The bundle that wants to get the status of the network interface (i.e. IP address, MAC address, subnet mask) will implement this interface and register it to the framework as OSGi service . The information corresponding to "type of network interface" is specified in service property `KEY_REQUESTED_NWIFTYPE` and "type of IP address" is specified in service property `KEY_REQUESTED_IPADDRESSTYPE` is notified.

1. In case that service property `KEY_REQUESTED_NWIFTYPE` and `KEY_REQUESTED_IPADDRESSTYPE` are specified to the OSGi service, it means that the registering bundle requires the callback of the list which includes all of the network interface information corresponding to the specified "type of network interface" and "type of IP address".
2. In case that service property `KEY_REQUESTED_NWIFTYPE` is specified to the OSGi service but service property `KEY_REQUESTED_IPADDRESSTYPE` is NOT specified to it, it means that the registering bundle requires the callback of the list which includes all of the network interface information corresponding to the specified "type of network interface".
3. In case that service property `KEY_REQUESTED_NWIFTYPE` is NOT specified to the OSGi service but service property `KEY_REQUESTED_IPADDRESSTYPE` is specified to it, the request from the registering bundle is considered illegal. Therefore, the registered service is not target of the callback.
4. In case that service property `KEY_REQUESTED_NWIFTYPE` and `KEY_REQUESTED_IPADDRESSTYPE` are NOT specified to the OSGi service, it means that the registering bundle requires the callback of the list which includes all network interface information.

NwIfInfo bundle monitors this OSGi service and calls back `statusChanged(NwIfInfoList)` in the two types of timing as below.

1. **When this OSGi service is registered**
   NwIfInfo bundle must call-back immediately the registering bundle with the information which is requested by the bundle and supported by NwIfInfo bundle as `NwIfInfoList`. In case that the multiple supported network interfaces are exist, `NwIfInfo` object of each network interface is created and added to `NwIfInfoList`. NwIfInfo bundle call-back the registering bundle with the `NwIfInfoList`. The unit of network interface is the same as unit of java.net.NetworkInterface object.
   In case that there is no network interface corresponding to the request when the service is registered, NwIfInfo bundle call-back the registering bundle with the empty `NwIfInfoList`.

2. **When the information of a network interface type or the status of IP address is associated with a network interface type is changed.**
   NwIfInfo bundle must call-back the registering bundle via `statusChanged(NwIfInfoList)` with the changed information which is requested by the bundle. This process is executed on one thread for one change of state. If there are multiple listeners for one change of status, the order of the callback is random order. However, regarding each changing of status, NwIfInfo bundle must call-back in chronological order of it.

NwIfinfo bundle must check whether the registering bundle has the appropriate `NwIfInfoListListenerPermission` which is described as below. If the bundle doesn't have the permission, NwIfinfo bundle must not call-back.

1. In case that `KEY_REQUESTED_NWIFTYPE` and `KEY_REQUESTED_IPADDRESSTYPE` are specified as service property, it is checked whether the registering bundle has the `NwIfInfoListListenerPermission` whose name is set ["type of network interface" + "." + "type of IP address"] that are specified as service property.
2. In case that `KEY_REQUESTED_NWIFTYPE` is specified but `KEY_REQUESTED_IPADDRESSTYPE` is NOT specified as service property, it is checked whether the registering bundle has the

NwIfInfoListListenerPermission whose name is set ["type of network interface" + "." + "*"] that is specified as service property.
3. In case that KEY_REQUESTED_NWIFTYPE is NOT specified but KEY_REQUESTED_IPADDRESSTYPE is specified as service property, the request from the registering bundle is considered illegal.
4. In case that KEY_REQUESTED_NWIFTYPE and KEY_REQUESTED_IPADDRESSTYPE are NOT specified as service property, it is checked whether the registering bundle has the NwIfInfoListListenerPermission whose name is set ["*"].

Example of registering this service:
In case that the bundles requests the network interface information corresponding to network interface type "WAN" or corresponding to network interface type "LAN" and IP address type "IPV4_PRIVATE".

```
        // The listener service requests information of network interface type "WAN".
        NwIfInfoListListener listenerA = new NwIfInfoListListenerImplA();
        Hashtable mapA = new Hashtable();
        mapA.put(NwIfInfoListListener.KEY_REQUESTED_NWIFTYPE, "WAN");
        bundleContext.registerService(NwIfInfoListListener.class.getName(), listenerA, mapA);

        // The listener service requests information of network interface type "LAN" and IP addr
ess type "IPV4_PRIVATE".
        NwIfInfoListListener listenerB = new NwIfInfoListListenerImplB();
        Hashtable mapB = new Hashtable();
        mapB.put(NwIfInfoListListener.KEY_REQUESTED_NWIFTYPE, "LAN");
        mapB.put(NwIfInfoListListener.KEY_REQUESTED_IPADDRESSTYPE, "IPV4_PRIVATE");
        bundleContext.registerService(NwIfInfoListListener.class.getName(), listenerB, mapB);
```

"network interface type" and "IP address type" which can be specified is defined as profile in another document. There is a limitation that "network interface type" and "IP address type" must not be included "." and "*".

Field Summary		Page
`String`	**KEY_REQUESTED_IPADDRESSTYPE** Key of service property to specify IP address type of requesting network information.	24
`String`	**KEY_REQUESTED_NWIFTYPE** Key of service property to specify network interface type of requesting network information.	24

Method Summary		Page
`void`	**statusChanged**(`NwIfInfoList nwIfInfoList`) The method to be notified the network interface information.	25

Field Detail

KEY_REQUESTED_NWIFTYPE

```
public static final String KEY_REQUESTED_NWIFTYPE = "requested.nwiftype"
```

Key of service property to specify network interface type of requesting network information.

KEY_REQUESTED_IPADDRESSTYPE

```
public static final String KEY_REQUESTED_IPADDRESSTYPE = "requested.ipaddresstype"
```

Key of service property to specify IP address type of requesting network information.

Method Detail

statusChanged

```
void statusChanged(NwIfInfoList nwIfInfoList)
```

The method to be notified the network interface information.
In case that there is no supported network interface information when the OSGi service is registered, the empty `NwIfInfoList` is called back.
This method should be return as soon as possible.

Parameters:

`nwIfInfoList` - List of network interface information.

Class NwIfInfoListListenerPermission

org.osgi.service.nwifinfo

```
java.lang.Object
  └java.security.Permission
      └java.security.BasicPermission
          └org.osgi.service.nwifinfo.NwIfInfoListListenerPermission
```

All Implemented Interfaces:

Guard, Serializable

```
public class NwIfInfoListListenerPermission
extends BasicPermission
```

This class represents access authority of the bundle which registers NwIfInfoListListener service. The NwIfInfoListListener service must be registered with service property including requested type of network interface and requested type of IP address. In case that the bundle has NwIfInfoListListenerPermission which is set corresponding type of network interface and type of IP address, the NwIfInfoListListener service can be called back with the required network interface information. NwIfInfoListListenerPermission uses only name argument which consists of "type of network interface" string and "type of IP address" string. The argument of actions is not used.

This class extends BasicPermission, BasicPermission.implies(java.security.Permission), BasicPermission.equals(Object), BasicPermission.hashCode() and BasicPermission.newPermissionCollection() don't need to be overridden in this class.

There are three types of name as below. "Str1" and "Str2" is used to represent string example. The length of Str1 and Str2 are greater or equal 1, "." and "*" are not included in the strings.

1. The name is consisted of Str1 and Str2 which is a dot-separated string (i.e. Str1.Str2). "*" is not included in the name.: "type of network interface" is described as Str1 and "type of IP address" is described as Str2. The bundle has the authority to get callback which includes the network interface information corresponding to "type of network interface" and "type of IP address" which are specified in the name.
2. The name is consisted of Str1 and "*" which is a dot-separated string (i.e. Str1.*).: "type of network interface" is described as Str1. The bundle has the authority to get callback which includes of the network interface information with the corresponding "type of network interface" which is specified in the name and all of "type of IP address".
3. The name is consisted of only "*".: The bundle has the authority to get callback which includes all of the network interface information.

Constructor Summary	Page
`NwIfInfoListListenerPermission(String name)` Constructor.	26

Constructor Detail

NwIfInfoListListenerPermission

```
public NwIfInfoListListenerPermission(String name)
```

Constructor.

Parameters:

`name` - The name of access authority.

8 Considered Alternatives

T.B.D

9 Security Considerations

T.B.D.

10 Document Support

10.1 References

[1]. Bradner, S., Key words for use in RFCs to Indicate Requirement Levels, RFC2119, March 1997.

[2]. Software Requirements & Specifications. Michael Jackson. ISBN 0-201-87712-0

10.2 Author's Address

Name	Shigekuni KONDO
Company	NTT Corporation
Address	1-1, Hikari-no-oka, Yokosuka-shi, 238-0847, Kanagawa, Japan
Voice	+81 46 859 3444
e-mail	kondo.shigekuni@lab.ntt.co.jp

10.3 Acronyms and Abbreviations

10.4 End of Document