



OSGiTM
Alliance

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

The OSGi Alliance hereby grants you a limited copyright license to copy and display this document (the "Distribution") in any medium without fee or royalty. This Distribution license is exclusively for the purpose of reviewing and providing feedback to the OSGi Alliance. You agree not to modify the Distribution in any way and further agree to not participate in any way in the making of derivative works thereof, other than as a necessary result of reviewing and providing feedback to the Distribution. You also agree to cause this notice, along with the accompanying consent, to be included on all copies (or portions thereof) of the Distribution. The OSGi Alliance also grants you a perpetual, non-exclusive, worldwide, fully paid-up, royalty free, limited license (without the right to sublicense) under any applicable copyrights, to create and/or distribute an implementation of the Distribution that: (i) fully implements the Distribution including all its required interfaces and functionality; (ii) does not modify, subset, superset or otherwise extend the OSGi Name Space, or include any public or protected packages, classes, Java interfaces, fields or methods within the OSGi Name Space other than those required and authorized by the Distribution. An implementation that does not satisfy limitations (i)-(ii) is not considered an implementation of the Distribution, does not receive the benefits of this license, and must not be described as an implementation of the Distribution. "OSGi Name Space" shall mean the public class or interface declarations whose names begin with "org.osgi" or any recognized successors or replacements thereof. The OSGi Alliance expressly reserves all rights not granted pursuant to these limited copyright licenses including termination of the license at will at any time.

EXCEPT FOR THE LIMITED COPYRIGHT LICENSES GRANTED ABOVE, THE OSGi ALLIANCE DOES NOT GRANT, EITHER EXPRESSLY OR IMPLIEDLY, A LICENSE TO ANY INTELLECTUAL PROPERTY IT, OR ANY THIRD PARTIES, OWN OR CONTROL. Title to the copyright in the Distribution will at all times remain with the OSGi Alliance. The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted therein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

THE DISTRIBUTION IS PROVIDED "AS IS," AND THE OSGi ALLIANCE (INCLUDING ANY THIRD PARTIES THAT HAVE CONTRIBUTED TO THE DISTRIBUTION) MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DISTRIBUTION ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.

NEITHER THE OSGi ALLIANCE NOR ANY THIRD PARTY WILL BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THE DISTRIBUTION.

Implementation of certain elements of this Distribution may be subject to third party intellectual property rights, including without limitation, patent rights (such a third party may or may not be a member of the OSGi Alliance). The OSGi Alliance is not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

The Distribution is a draft. As a result, the final product may change substantially by the time of final publication, and you are cautioned against relying on the content of this Distribution. You are encouraged to update any implementation of the Distribution if and when such Distribution becomes a final specification.

The OSGi Alliance is willing to receive input, suggestions and other feedback ("Feedback") on the Distribution. By providing such Feedback to the OSGi Alliance, you grant to the OSGi Alliance and all its Members a non-exclusive, non-transferable,

worldwide, perpetual, irrevocable, royalty-free copyright license to copy, publish, license, modify, sublicense or otherwise distribute and exploit your Feedback for any purpose. Likewise, if incorporation of your Feedback would cause an implementation of the Distribution, including as it may be modified, amended, or published at any point in the future ("Future Specification"), to necessarily infringe a patent or patent application that you own or control, you hereby commit to grant to all implementers of such Distribution or Future Specification an irrevocable, worldwide, sublicenseable, royalty free license under such patent or patent application to make, have made, use, sell, offer for sale, import and export products or services that implement such Distribution or Future Specification. You warrant that (a) to the best of your knowledge you have the right to provide this Feedback, and if you are providing Feedback on behalf of a company, you have the rights to provide Feedback on behalf of your company; (b) the Feedback is not confidential to you and does not violate the copyright or trade secret interests of another; and (c) to the best of your knowledge, use of the Feedback would not cause an implementation of the Distribution or a Future Specification to necessarily infringe any third-party patent or patent application known to you. You also acknowledge that the OSGi Alliance is not required to incorporate your Feedback into any version of the Distribution or a Future Specification.

I HEREBY ACKNOWLEDGE AND AGREE TO THE TERMS AND CONDITIONS DELINEATED ABOVE.

0.2 Trademarks

OSGi™ is a trademark, registered trademark, or service mark of the OSGi Alliance in the US and other countries. Java is a trademark, registered trademark, or service mark of Oracle Corporation in the US and other countries. All other trademarks, registered trademarks, or service marks used in this document are the property of their respective owners and are hereby recognized.

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 Document Information.....	2
0.1 License.....	2
0.2 Trademarks.....	3
0.3 Feedback.....	3
0.4 Table of Contents.....	3
0.5 Terminology and Document Conventions.....	4
0.6 Revision History.....	4
 1 Introduction.....	 5
 2 Application Domain.....	 5
 3 Problem Description.....	 6
 4 Requirements.....	 7
 5 Technical Solution.....	 8
5.1 NwIfInfo.....	9
5.2 NwIfInfoAdmin.....	9
5.3 NwIfInfoList.....	9
5.4 NwIfInfoListListener	9
5.5 NwIfInetAddress.....	9
5.6 NwInetAddressList.....	9
5.7 NwIfInfoListListenerPermission.....	10

6 Data Transfer Objects.....	10
7 Javadoc.....	10
8 Considered Alternatives.....	27
9 Security Considerations.....	27
10 Document Support.....	28
10.1 References.....	28
10.2 Author's Address.....	28
10.3 Acronyms and Abbreviations.....	28
10.4 End of Document.....	28

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 particular, 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.

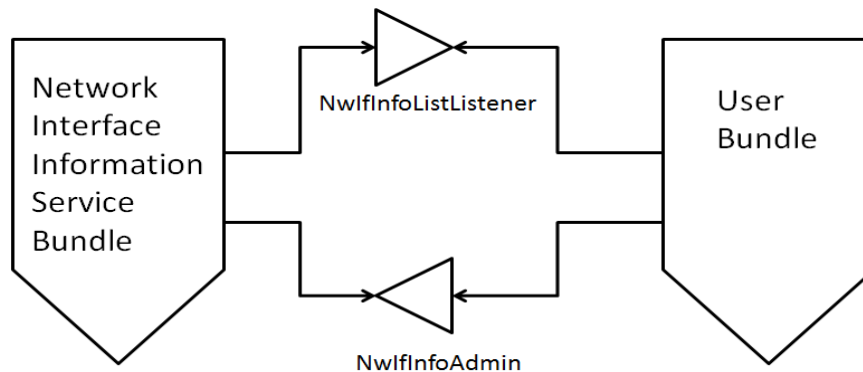
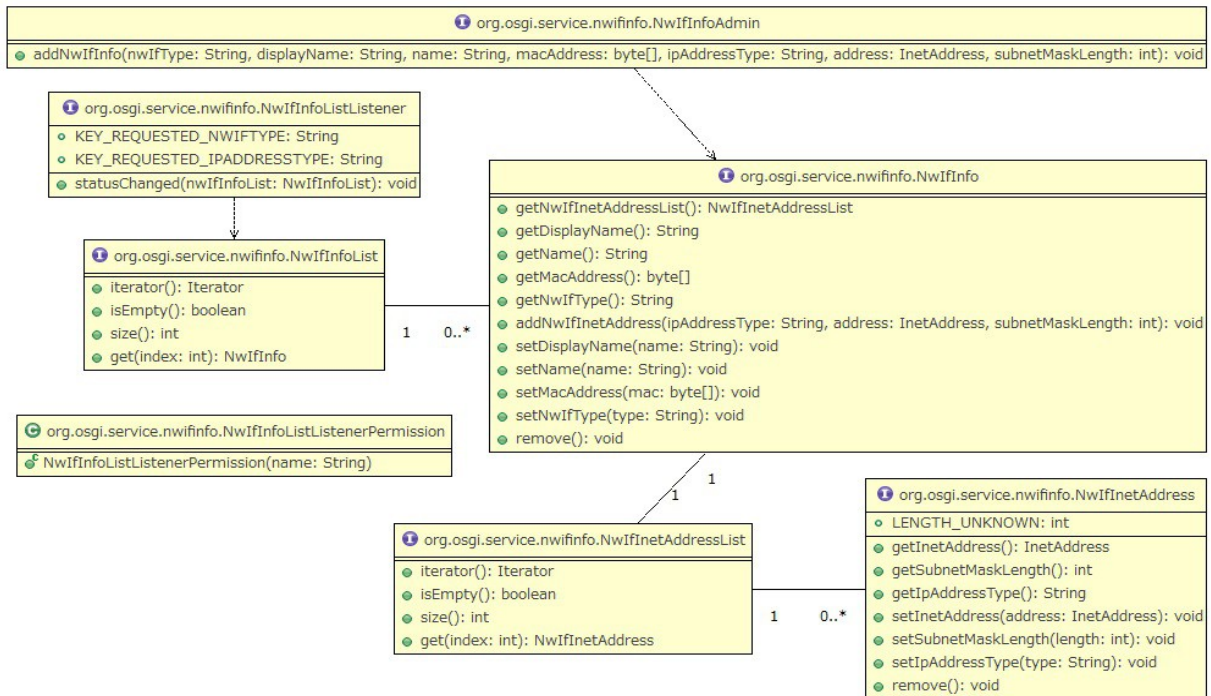
[REQ_6] The solution MUST support both IPv4 and IPv6 environments (mixed or separately) and the corresponding characteristics, for example IPv4 and IPv6 addresses, multi-prefixes, multicast etc. .

[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



5.1 NwlflInfo

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 NwlflInfoAdmin

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

5.3 NwlflInfoList

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

5.4 NwlflInfoListListener

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 NwlflnetAddress

This interface represents an IP address information.

5.6 NwlInetAddressList

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

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

5.7 NwlfnfoListListenerPermission

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

The NwlfnfoListListener 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 NwlfnfoListListenerPermission which is set corresponding type of network interface and type of IP address, the NwlfnfoListListener 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
<u>NwlflnetAddresses</u>	This interface represents an IP address information.	13
<u>NwlflnetAddressesList</u>	This Interface represents a list of IP addresses for network IF.	15
<u>NwlflInfo</u>	This interface represents network interface information.	17
<u>NwlflInfoAdmin</u>		20
<u>NwlflInfoList</u>	This interface represents a list of network IF information.	21
<u>NwlflInfoListListener</u>	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
<u>NwlflInfoListListenerPermission</u>	This class represents access authority of the bundle which registers NwlflInfoListListener 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

getNwlftype

String **getNwlftype**()

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

addNwlfinetAddress

```
void addNwlfinetAddress(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		Pag e
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 address 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.

Java API documentation generated with [DocFlex/Doclet](#) v1.5.6

DocFlex/Doclet is both a multi-format Javadoc doclet and a free edition of [DocFlex/Javadoc](#). If you need to customize your Javadoc without writing a full-blown doclet from scratch, DocFlex/Javadoc may be the only tool able to help you! Find out more at www.docflex.com

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