RFC 209 – Network Interface Information Service

Draft

32 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

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

0.5 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
0.2	Feb 10, 2014	Based on the last meeting, the section 5 has changed. Changed the design to service repository model. Shigekuni Kondo, NTT Corporation, kondo.shigekuni@lab.ntt.co.jp

Revision	Date	Comments
0.3	Feb 28, 2014	Based on the last meeting, the following points have been modified. nwif.disprayname is changed to OPTIONAL. Interface name is changed. NwIfInfo --> NetworkAdapter, NwIfInetAddress --> NetworkAddress IPAdress Type is divided to IPAdresVersion and IPAdressScope. The functionality of configuration is removed. Shigekuni Kondo, NTT Corporation, kondo.shigekuni@lab.ntt.co.jp
0.4	Mar 6, 2014	Based on the last meeting, the following points have been modified. Update IPAdress Version and Scope. The order of registering (unregistering) NetworkAdapter and NetworkAddress is changed. Shigekuni Kondo, NTT Corporation, kondo.shigekuni@lab.ntt.co.jp
0.5	Mar 14, 2014	Based on the last confference call, the following points have been modified. Adding new service properties to NetworkAdapter service and NetworkAddress. Some service property names are modified. NetworkAadapterException class is removed. JavaDoc update. Some sentences are modified. Shigekuni Kondo, NTT Corporation, kondo.shigekuni@lab.ntt.co.jp
0.6	Mar 19, 2014	Fixed some sentences based on the comments from Evgeni. Added Chapter 8 Considered Alternatives section. Shigekuni Kondo, NTT Corporation, kondo.shigekuni@lab.ntt.co.jp
0.7	Apr 9, 2014	Fixed the packagename (Java doc). Added Chapter 9 Added Security section. Some sentences are modified. 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

5.1   Intoroduction

When the IP address is changed, the bundles utilize IP address information (i.e. Http Service bundle running HTTP Servers) is necessary to detect the fact of the change. In case of using the standard Java API, such as java.net.InetAddress and java.net.NetworkInterface, processing to confirm the IP address at regular intervals from the bundle itself is required. Since this is a process common to all bundles which are necessary to detect the change of IP address information, provision of services to notify a change of IP address is very effective.

Therefore the API provides the change notification feature for each network interface information (including the IP address information) is investigated in this RFC document. In addition to that, this RFC defines APIs which provide the functionalities to obtain the network interface information and the information of IP address which is bound to the network interface .

The name of the network interface is information dependent on the operating system. In order to be able to bundle implementation that uses the Network Interface Information Service is not aware of the differences in the operating system, the mechanism of identifying network interface is necessary in a format that does not depend on the operating system. This RFC also defines it.

5.2   Entities

- Network Interface
  Available and activated network interfaces provided in the execution environment. In this specification, the unit of the network interface is the logical interface, not the physical interface.

- NetworkAdapter
  The OSGi service that provides information related to the Network Interface. This service provides functionalities corresponding to "java.net.NetworkInterface".

- NetworkAddress
  The OSGi service that provides information of IP addresses available on execution environment on a Network Interface Information Service bundle is running.

- NetworkAdapterType
  The identifier of the network interface to be defined in a manner. It is independent of the operating system. This identifier string is not specified in this specification. The Network Interface Information Service bundle provider should define this identifier string. This identifier is used by user bundle to specify the network interface to be monitored.

- IPAdressVersion
  An identifier indicating the version of IP address (i.e. Ipv4, IPv6). This identifier is defined in this specification. This identifier is used by user bundle to specify the network interface to be monitored.

- IPAdressScope
An identifier indicating the scope of IP address (i.e. GROVAL, PRIVATE). This identifier is defined in this specification. This identifier is used by user bundle to specify the network interface to be monitored.
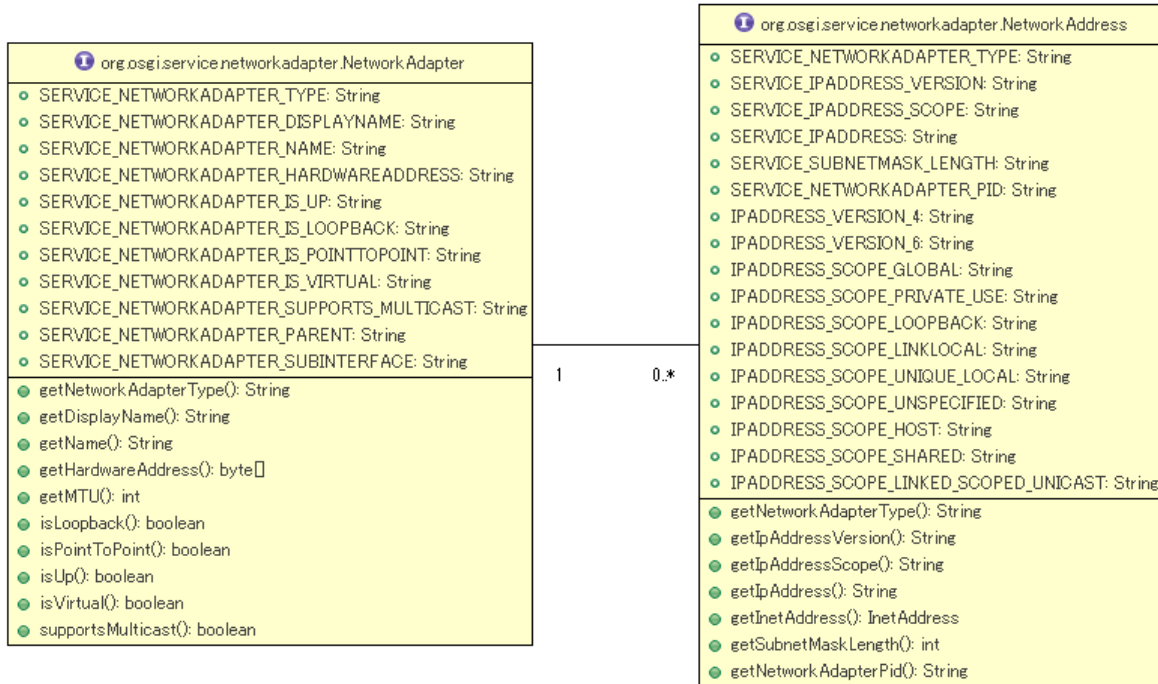
Fig.1 Class structure of Network Interface Information Service

<Network Interface Information Bundle>

To register two kind of services.

NetworkAdapter service provides network interface information, this bundle registers this service each logical interface.

NetworkAddress service provides each IP address information, this bundle registers this service each IP address.

NetworkAddress service is associated with specific NetworkAdapter service.

When information of network interface is changed, service properties of NetworkAdapter setvice and NetworkAddress service will be modified.

<User bundle>

Tracking necessary NetworkAdapter service and NetworkAddress service (using filter).This bundle can be notified the change of network interface information via Service Event.
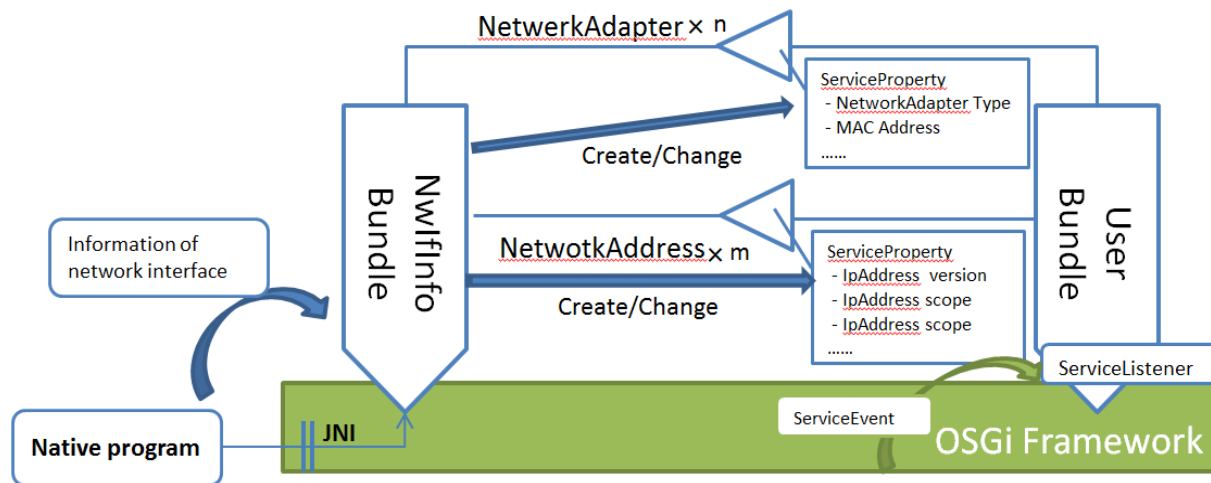
Fig.2 Overview of Network Interface Information Service

5.3 NetworkAdapter Service

NetworkAdapter is an interface that provides information about single network interfaces that are provided by the execution environment. In case that multiple network interfaces are present, NetworkAdapter Services that corresponds to each network interface must be registered.

NetworkAdapter service is registered to the service repository with service properties which are shown in the following table.

Table 1. Service properties of NetworkAdapter Service

The key of service property	Description
service.networkAdapter.type	Required property. Network interface type is set to a value.
service.networkAdapter.hardwareAddress	Required property. Hardware address (MAC address) is set to a value. This property can also be obtained from getHardwareAddress().
service.networkAdapter.name	Required property. Network interface name is set to a value. This property can also be obtained from getName().
service.networkAdapter.displayName	Optional property. Network interface display name is set to a value. This property can also be obtained from getDisplayName().

service.networkAdapter.isUp	Optional property. The value is true when a network interface is up and running, otherwise it is false.
service.networkAdapter.isLoopback	Optional property. The value is true when a network interface is a loopback interface, otherwise it is false.
service.networkAdapter.isPointToPoint	Optional property. The value is true when a network interface is a point to point interface, otherwise it is false.
service.networkAdapter.isVirtual	Optional property. The value is true when a network interface is a virtual interface, otherwise it is false.
service.networkAdapter.supportsMulticast	Optional property. The value is true when a network interface supports multicasting, otherwise it is false.
service.networkAdapter.parent	Optional property. Service PID of the NetworkAdapter service which is parent of this NetworkAdapter is specified.
service.networkAdapter.subInterface	Optional property. Service PID of the NetworkAdapter service which is subinterface of this NetworkAdapter is specified.

When a network interface becomes available, NetworkAdapter service associated with the network interface is registered to service repository. If the network interface becomes unavailable, the corresponding NetworkAdapter service is unregistered.

When the attribute values of the network interface are set to the service property changes, NetworkAdapter service is updated. NetworkAdapter interface provides a method corresponding to java.net.NetworkInterface in order to provide information on the network interface associated with. However, it does not provide in this interface method corresponding to the Static method. In addition to that, because NetworkInterface object or InetAddress object is registered in the service repository as NetworkAdapter and NetworkAdress, the method to get those objects does not provide in NetworkAdapter interface. NetworkAdapter provides a method to retrieve the value of an attribute of a network interface.

Table 2. Investigation on the method to be adopted based on Java standard API

Method in java.net.NetworkInterface	Adoption
getByInetAddress(InetAddress)	Not adopted in this interface because NetworkAdapter service is registered.
getByName(String)	Not adopted in this interface because NetworkAdapter service is registered.
getDisplayName()	Adopted in this interface.
getHardwareAddress()	Adopted in this interface.
getInetAddresses()	Not adopted in this interface because InetAdress object is provided

	by NetworkAdress service.
getInterfaceAddresses()	Not adopted in this interface because InetAdress object is provided by NetworkAdress service.
getMTU()	Adopted in this interface.
getName()	Adopted in this interface.
getNetworkInterfaces()	Not adopted in this interface because NetworkAdapter service is registered.
getParent()	Not adopted in this interface because NetworkAdapter service is registered.
getSubInterfaces()	Not adopted in this interface because NetworkAdapter service is registered.
isLoopback()	Adopted in this interface as service property.
isPointToPoint()	Adopted in this interface as service property.
isUp()	Adopted in this interface as service property.
isVirtual()	Adopted in this interface as service property.
supportsMulticast()	Adopted in this interface as service property.

5.4   NetworkAddress Service

NetworkAddress interface provides information of IP addresses available on execution environment on a Network Interface Information Service bundle is running.

NetworkAddress service is registered to the service repository with service properties which are shown in the following table.

Table 3. Service properties of NetworkAddress Service

The key of service property	Description
service.networkAdapter.type	Required property. Network interface type is set to a value.
service.ipAddress.version	Required property. IP address version is set to a value.
service.ipAddress.scope	Required property. IP address scope is set to a value.
service.ipAddress	Required property. IP address String is set to a

	value.
service.subnetmask.length	Required property. subnet mask length of the required properties IPv4, or IPv6 prefix length is set to a value.
service.networkAdapter.pid	Required property. Service ID of the NetworkAdapter servuice corresponding to the network interface binding this IP address is set to a value.

NetworkAddress service is registered to service repository for each available IP address.

When the associated IP addresses is deleted, or the network interface that the IP address is bound becomes unavailable, the NetworkAddress service is unregistered. When the associated IP address has been changed, NetworkAddress service is updated. The user bundle can detect the change of IP address by monitoring the registration or unregistering, updating of NetworkAddress service.

Because IP addresses are bound to the network interface of any, Service PID of the associated NetworkAdapter service and its network interface type are set to service property.

NetworkAdapter service MUST be registered after the all associated NetworkAddress services are registered. On the other hand, in case of unregistering services, after associated NetworkAdapter service is unregistered, NetworkAddress all related services are unregistered..

5.5　Network interface type and IP address type
5.5.1　Network interface type

In order to identify the network interface, it is possible to use the network interface name.

However, since the network interface name is an identifier that is dependent on the operating system, if network interface name is used as identifier, it is necessary to implement the user bundle being aware of the operating system. Therefore, in this specification, "network interface type" which is independent of the operating system is used to identify the network interface. The network interface type sting itself is not defined in this specification. It should be provided by the platform provider on which Network Interface Information Service bundle is running. For example, Network interface type "LAN" indicates the network interface to connect to a local area network, Network interface type "WAN" indicates the network interface to connect to the Internet. If a bundle wants to obtain the information of the network interface which connects to the Internet, the bundle is able to get it to obtain NetworkAdapter service which is set "SERVICE_NETWORKADAPTER_TYPE = WAN" to service property from service repository.

5.5.2　IP address type

This spec defines "IP address version" and "IP address scope" as IP address type to be narrowed down the IP address by user bunde of following.

Table 4. IP Address Version

IP Address Version	Discription
IPV4	IP address version which means IPv4 address.
IPV6	IP address version which means IPv6 address.

Table 5. IP Address Scope (T.B.D)

IP Address Scope	Description
GLOBAL	IP address scope which means global address.
PRIVATE_USE	IP address scope which means private address.
LOOPBACK	IP address scope which means loopback address.
LINKLOCAL	IP address scope which means linklocal address.
UNIQUE_LOCAL	IP address scope which means unique-local address.
UNSPECIFIED	IP address scope which means the absence of an address.

If a bundle which wants to check for the IP address of the IPv4 global, the bundle is able to confirm to obtain NetworkAddress service which is set "SERVICE_IPADDRESS_VERSION = IPV4" and "SERVICE_IPADDRESS_SCOPE = GLOBAL" to service property from service repository.

6 Data Transfer Objects

This RFC will not provide Data Transfer Objects.

7 Javadoc

OSGi Javadoc

14/04/02 14:30

Package Summary	Page
org.osgi.service.networkadapter	*18*

Package org.osgi.service.networkadapter

Interface Summary		Page
NetworkAdapter	This interface represents network interface information.	*19*
NetworkAddress	This interface represents an IP address information.	*25*

Interface NetworkAdapter

org.osgi.service.networkadapter

---

public interface **NetworkAdapter**

This interface represents network interface information. Network interface information service is set the following information as service property.

1. SERVICE_NETWORKADAPTER_TYPE : Network Interface Type
2. SERVICE_NETWORKADAPTER_DISPLAYNAME : Network Interface Display Name
3. SERVICE_NETWORKADAPTER_NAME : Network Interface Name
4. SERVICE_HARDWAREADDRESS : Hardware Address

Field Summary		Page
String	**SERVICE_NETWORKADAPTER_DISPLAYNAME** The key string of "service.networkAdapter.displayName" service property.	*20*
String	**SERVICE_NETWORKADAPTER_HARDWAREADDRESS** The key string of "service.networkAdapter.hardwareAddress" service property.	*20*
String	**SERVICE_NETWORKADAPTER_IS_LOOPBACK** The key string of "service.networkAdapter.isLoopback" service property.	*21*
String	**SERVICE_NETWORKADAPTER_IS_POINTTOPOINT** The key string of "service.networkAdapter.isPointToPoint" service property.	*21*
String	**SERVICE_NETWORKADAPTER_IS_UP** The key string of "service.networkAdapter.isUp" service property.	*21*
String	**SERVICE_NETWORKADAPTER_IS_VIRTUAL** The key string of "service.networkAdapter.isVirtual" service property.	*21*
String	**SERVICE_NETWORKADAPTER_NAME** The key string of "service.networkAdapter.name" service property.	*20*
String	**SERVICE_NETWORKADAPTER_PARENT** The key string of "service.networkAdapter.parent" service property.	*21*
String	**SERVICE_NETWORKADAPTER_SUBINTERFACE** The key string of "service.networkAdapter.subInterface" service property.	*22*
String	**SERVICE_NETWORKADAPTER_SUPPORTS_MULTICAST** The key string of "service.networkAdapter.supportsMulticast" service property.	*21*
String	**SERVICE_NETWORKADAPTER_TYPE** The key string of "service.networkAdapter.type" service property.	*20*

Method Summary		Page
String	**getDisplayName**() Returns the network interface display name of "service.networkAdapter.displayname" service property value.	*22*
byte[]	**getHardwareAddress**() Returns the MAC address of "service.networkAdapter.hardwareAddress" service property value.	*22*
int	**getMTU**() Returns the Maximum Transmission Unit (MTU) of this interface.	*23*

String	**getName**() Returns the network interface name of "service.networkAdapter.name" service property value.	*22*
String	**getNetworkAdapterType**() Returns the network interface type of "service.networkAdapter.type" service property value.	*22*
boolean	**isLoopback**() Returns whether a network interface is a loopback interface.	*23*
boolean	**isPointToPoint**() Returns whether a network interface is a point to point interface.	*23*
boolean	**isUp**() Returns whether a network interface is up and running.	*23*
boolean	**isVirtual**() Returns whether this interface is a virtual interface (also called subinterface).	*24*
boolean	**supportsMulticast**() Returns whether a network interface supports multicasting or not.	*24*

Field Detail

SERVICE_NETWORKADAPTER_TYPE

```
public static final String SERVICE_NETWORKADAPTER_TYPE = "service.networkAdapter.type"
```

The key string of "service.networkAdapter.type" service property. Network Interface Type is specified.

SERVICE_NETWORKADAPTER_DISPLAYNAME

```
public static final String SERVICE_NETWORKADAPTER_DISPLAYNAME = "service.networkAdapter.displayName"
```

The key string of "service.networkAdapter.displayName" service property. Network Interface Display Name is specified.

SERVICE_NETWORKADAPTER_NAME

```
public static final String SERVICE_NETWORKADAPTER_NAME = "service.networkAdapter.name"
```

The key string of "service.networkAdapter.name" service property. Network Interface Name is specified.

SERVICE_NETWORKADAPTER_HARDWAREADDRESS

```
public static final String SERVICE_NETWORKADAPTER_HARDWAREADDRESS = "service.networkAdapter.hardwareAddress"
```

The key string of "service.networkAdapter.hardwareAddress" service property. Hardware Address is specified.

SERVICE_NETWORKADAPTER_IS_UP

```
public static final String SERVICE_NETWORKADAPTER_IS_UP = "service.networkAdapter.isUp"
```

The key string of "service.networkAdapter.isUp" service property. The value is true when a network interface is up and running, otherwise it is false.

SERVICE_NETWORKADAPTER_IS_LOOPBACK

```
public static final String SERVICE_NETWORKADAPTER_IS_LOOPBACK = "service.networkAdapter.isLoopback"
```

The key string of "service.networkAdapter.isLoopback" service property. The value is true when a network interface is a loopback interface, otherwise it is false.

SERVICE_NETWORKADAPTER_IS_POINTTOPOINT

```
public static final String SERVICE_NETWORKADAPTER_IS_POINTTOPOINT = "service.networkAdapter.isPointToPoint"
```

The key string of "service.networkAdapter.isPointToPoint" service property. The value is true when a network interface is a point to point interface, otherwise it is false.

SERVICE_NETWORKADAPTER_IS_VIRTUAL

```
public static final String SERVICE_NETWORKADAPTER_IS_VIRTUAL = "service.networkAdapter.isVirtual"
```

The key string of "service.networkAdapter.isVirtual" service property. The value is true when a network interface is a virtual interface, otherwise it is false.

SERVICE_NETWORKADAPTER_SUPPORTS_MULTICAST

```
public static final String SERVICE_NETWORKADAPTER_SUPPORTS_MULTICAST = "service.networkAdapter.supportsMulticast"
```

The key string of "service.networkAdapter.supportsMulticast" service property. The value is true when a network interface supports multicasting, otherwise it is false.

SERVICE_NETWORKADAPTER_PARENT

```
public static final String SERVICE_NETWORKADAPTER_PARENT = "service.networkAdapter.parent"
```

The key string of "service.networkAdapter.parent" service property.
Service PID of the NetworkAdapter service which is parent of this NetworkAdapter is specified.

SERVICE_NETWORKADAPTER_SUBINTERFACE

public static final String **SERVICE_NETWORKADAPTER_SUBINTERFACE** = "service.networkAdapter.subInterface"

The key string of "service.networkAdapter.subInterface" service property.
Service PID of the NetworkAdapter service which is subinterface of this NetworkAdapter is specified.

Method Detail

getNetworkAdapterType

String **getNetworkAdapterType**()

Returns the network interface type of "service.networkAdapter.type" service property value.

Returns:
Network Interface Type

---

getDisplayName

String **getDisplayName**()

Returns the network interface display name of "service.networkAdapter.displayname" service property value.

Returns:
Network Interface Display Name

---

getName

String **getName**()

Returns the network interface name of "service.networkAdapter.name" service property value.

Returns:
Network Interface Name

---

getHardwareAddress

byte[] **getHardwareAddress**()

Returns the MAC address of "service.networkAdapter.hardwareAddress" service property value.

Returns:
Hardware Address

---

getMTU

```
int getMTU()
    throws SocketException
```

Returns the Maximum Transmission Unit (MTU) of this interface.

Returns:
The value of the MTU for that interface.

Throws:
SocketException - If an I/O error occurs.

---

isLoopback

```
boolean isLoopback()
        throws SocketException
```

Returns whether a network interface is a loopback interface.

Returns:
true if the interface is a loopback interface.

Throws:
SocketException - If an I/O error occurs.

---

isPointToPoint

```
boolean isPointToPoint()
            throws SocketException
```

Returns whether a network interface is a point to point interface.

Returns:
true if the interface is a point to point interface.

Throws:
SocketException - If an I/O error occurs.

---

isUp

```
boolean isUp()
      throws SocketException
```

Returns whether a network interface is up and running.

Returns:
true if the interface is up and running.

Throws:
SocketException - If an I/O error occurs.

isVirtual

```
boolean isVirtual()
```

Returns whether this interface is a virtual interface (also called subinterface). Virtual interfaces are, on some systems, interfaces created as a child of a physical interface and given different settings (like address or MTU). Usually the name of the interface will the name of the parent followed by a colon (:) and a number identifying the child since there can be several virtual interfaces attached to a single physical interface.

Returns:
true if this interface is a virtual interface.

supportsMulticast

```
boolean supportsMulticast()
              throws SocketException
```

Returns whether a network interface supports multicasting or not.

Returns:
true if the interface supports Multicasting.

Throws:
SocketException - If an I/O error occurs.

Interface NetworkAddress

org.osgi.service.networkadapter

---

public interface **NetworkAddress**

This interface represents an IP address information.
IP address information service is set the following information as service property.

1. SERVICE_NETWORKADAPTER_TYPE : Network Interface Type
2. SERVICE_IPADDRESS_VERSION : IP Address Version
3. SERVICE_IPADDRESS_SCOPE : IP Address Scope
4. SERVICE_IPADDRESS : IP Address
5. SERVICE_SUBNETMASK_LENGTH : Subnet Mask Length(IPv4) or Prefix Length(IPv6)
6. SERVICE_NETWORKADAPTER_PID : Service PID of the NetworkAdapter service to which this service belongs

---

Field Summary		*Page*
String	**IPADDRESS_SCOPE_GLOBAL** The string of IP address scope which means global address.	*27*
String	**IPADDRESS_SCOPE_HOST** The string of IP address scope which means "This host on this network".	*28*
String	**IPADDRESS_SCOPE_LINKED_SCOPED_UNICAST** The string of IP address scope which means "Linked-Scoped Unicast".	*28*
String	**IPADDRESS_SCOPE_LINKLOCAL** The string of IP address scope which means "Link Local".	*28*
String	**IPADDRESS_SCOPE_LOOPBACK** The string of IP address scope which means "Loopback".	*27*
String	**IPADDRESS_SCOPE_PRIVATE_USE** The string of IP address scope which means "Private-Use Networks".	*27*
String	**IPADDRESS_SCOPE_SHARED** The string of IP address scope which means "Shared Address Space".	*28*
String	**IPADDRESS_SCOPE_UNIQUE_LOCAL** The string of IP address scope which means "Unique-Local".	*28*
String	**IPADDRESS_SCOPE_UNSPECIFIED** The string of IP address scope which means "Unspecified Address".	*28*
String	**IPADDRESS_VERSION_4** The string of IP address version which means IP address version 4.	*27*
String	**IPADDRESS_VERSION_6** The string of IP address version which means IP address version 6.	*27*
String	**SERVICE_IPADDRESS** The key string of "service.ipAddress" service property.	*26*
String	**SERVICE_IPADDRESS_SCOPE** The key string of "service.ipAddress.scope" service property.	*26*
String	**SERVICE_IPADDRESS_VERSION** The key string of "service.ipAddress.version" service property.	*26*
String	**SERVICE_NETWORKADAPTER_PID** The key string of "service.networkAdapter.id" service property.	*27*
String	**SERVICE_NETWORKADAPTER_TYPE** The key string of "service.networkAdapter.type" service property.	*26*

`String`	**SERVICE_SUBNETMASK_LENGTH** The key string of "service.subnetmask.length" service property.	*27*

Method Summary		*Pag e*
`InetAddres s`	**getInetAddress**() Returns the InetAddress object of this IP address.	*29*
`String`	**getIpAddress**() Returns the IP address of "service.ipaddress" service property value.	*29*
`String`	**getIpAddressScope**() Returns the IP address scope of "service.ipaddress.scope" service property value.	*29*
`String`	**getIpAddressVersion**() Returns the IP address version of "service.ipaddress.version" service property value.	*29*
`String`	**getNetworkAdapterPid**() Returns the "service.networkadapter.pid" service property value.	*30*
`String`	**getNetworkAdapterType**() Returns the network interface type of "service.networkAdapter.type" service property value.	*29*
`int`	**getSubnetMaskLength**() Returns the "service.subnetmask.length" service property value.	*30*

Field Detail

SERVICE_NETWORKADAPTER_TYPE

`public static final String` **SERVICE_NETWORKADAPTER_TYPE** `= "service.networkAdapter.type"`

> The key string of "service.networkAdapter.type" service property. Network Interface Type is specified.

SERVICE_IPADDRESS_VERSION

`public static final String` **SERVICE_IPADDRESS_VERSION** `= "service.ipAddress.version"`

> The key string of "service.ipAddress.version" service property. IP Address Type is specified.

SERVICE_IPADDRESS_SCOPE

`public static final String` **SERVICE_IPADDRESS_SCOPE** `= "service.ipAddress.scope"`

> The key string of "service.ipAddress.scope" service property. IP Address Type is specified.

SERVICE_IPADDRESS

`public static final String` **SERVICE_IPADDRESS** `= "service.ipAddress"`

> The key string of "service.ipAddress" service property. IP Address is specified.

SERVICE_SUBNETMASK_LENGTH

public static final String **SERVICE_SUBNETMASK_LENGTH** = "service.subnetmask.length"

The key string of "service.subnetmask.length" service property. Subnet Mask Length(IPv4) or Prefix Length(IPv6) is specified.

SERVICE_NETWORKADAPTER_PID

public static final String **SERVICE_NETWORKADAPTER_PID** = "service.networkAdapter.pid"

The key string of "service.networkAdapter.id" service property. Service PID of the interface information service to which it belongs is specified.

IPADDRESS_VERSION_4

public static final String **IPADDRESS_VERSION_4** = "IPV4"

The string of IP address version which means IP address version 4.

IPADDRESS_VERSION_6

public static final String **IPADDRESS_VERSION_6** = "IPV6"

The string of IP address version which means IP address version 6.

IPADDRESS_SCOPE_GLOBAL

public static final String **IPADDRESS_SCOPE_GLOBAL** = "GLOBAL"

The string of IP address scope which means global address. The global address is defined as the address other than the address defined in the RFC6890.

IPADDRESS_SCOPE_PRIVATE_USE

public static final String **IPADDRESS_SCOPE_PRIVATE_USE** = "PRIVATE_USE"

The string of IP address scope which means "Private-Use Networks". See RFC6890 for the definition of "Private-Use Networks".

IPADDRESS_SCOPE_LOOPBACK

public static final String **IPADDRESS_SCOPE_LOOPBACK** = "LOOPBACK"

The string of IP address scope which means "Loopback".
See RFC6890 for the definition of "Loopback".

IPADDRESS_SCOPE_LINKLOCAL

`public static final String` **`IPADDRESS_SCOPE_LINKLOCAL`** `= "LINKLOCAL"`

The string of IP address scope which means "Link Local".
See RFC6890 for the definition of "Link Local".

IPADDRESS_SCOPE_UNIQUE_LOCAL

`public static final String` **`IPADDRESS_SCOPE_UNIQUE_LOCAL`** `= "UNIQUE_LOCAL"`

The string of IP address scope which means "Unique-Local".
See RFC6890 for the definition of "Unique-Local".

IPADDRESS_SCOPE_UNSPECIFIED

`public static final String` **`IPADDRESS_SCOPE_UNSPECIFIED`** `= "UNSPECIFIED"`

The string of IP address scope which means "Unspecified Address".
See RFC6890 for the definition of "Unspecified Address".

IPADDRESS_SCOPE_HOST

`public static final String` **`IPADDRESS_SCOPE_HOST`** `= "HOST"`

The string of IP address scope which means "This host on this network".
See RFC6890 for the definition of "This host on this network".

IPADDRESS_SCOPE_SHARED

`public static final String` **`IPADDRESS_SCOPE_SHARED`** `= "SHARED"`

The string of IP address scope which means "Shared Address Space".
See RFC6890 for the definition of "Shared Address Space".

IPADDRESS_SCOPE_LINKED_SCOPED_UNICAST

`public static final String` **`IPADDRESS_SCOPE_LINKED_SCOPED_UNICAST`** `= "LINKED_SCOPED_UNICAST"`

The string of IP address scope which means "Linked-Scoped Unicast".
See RFC6890 for the definition of "Linked-Scoped Unicast".

Method Detail

getNetworkAdapterType

```
String getNetworkAdapterType()
```

Returns the network interface type of "service.networkAdapter.type" service property value.

Returns:
Network Interface Type

---

getIpAddressVersion

```
String getIpAddressVersion()
```

Returns the IP address version of "service.ipaddress.version" service property value.

Returns:
IP Address Version

---

getIpAddressScope

```
String getIpAddressScope()
```

Returns the IP address scope of "service.ipaddress.scope" service property value.

Returns:
IP Address Scope

---

getIpAddress

```
String getIpAddress()
```

Returns the IP address of "service.ipaddress" service property value.

Returns:
IP Address string

---

getInetAddress

```
InetAddress getInetAddress()
```

Returns the InetAddress object of this IP address. Returned object is created from "service.ipaddress" service property value.

Returns:
InetAddress

getSubnetMaskLength

```
int getSubnetMaskLength()
```

Returns the "service.subnetmask.length" service property value.

Returns:
Subnet Mask Length(IPv4) or Prefix Length(IPv6)

getNetworkAdapterPid

```
String getNetworkAdapterPid()
```

Returns the "service.networkadapter.pid" service property value.

Returns:
Service ID of the interface information service to which it belongs

---

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

8.1 Whiteboard pattern model

<NwInfo Bundle>

To provide (export) Listener service interface. This bundle gets the Listener services. When information of network interface is changed, this bundle prepares List of network interface information and call back the Listener services. This bundle will provide filter mechanism. User bundle can get only the necessary information to use the functionality.

<User bundle>

To register the Listener service, and waiting for notification of network interface information.
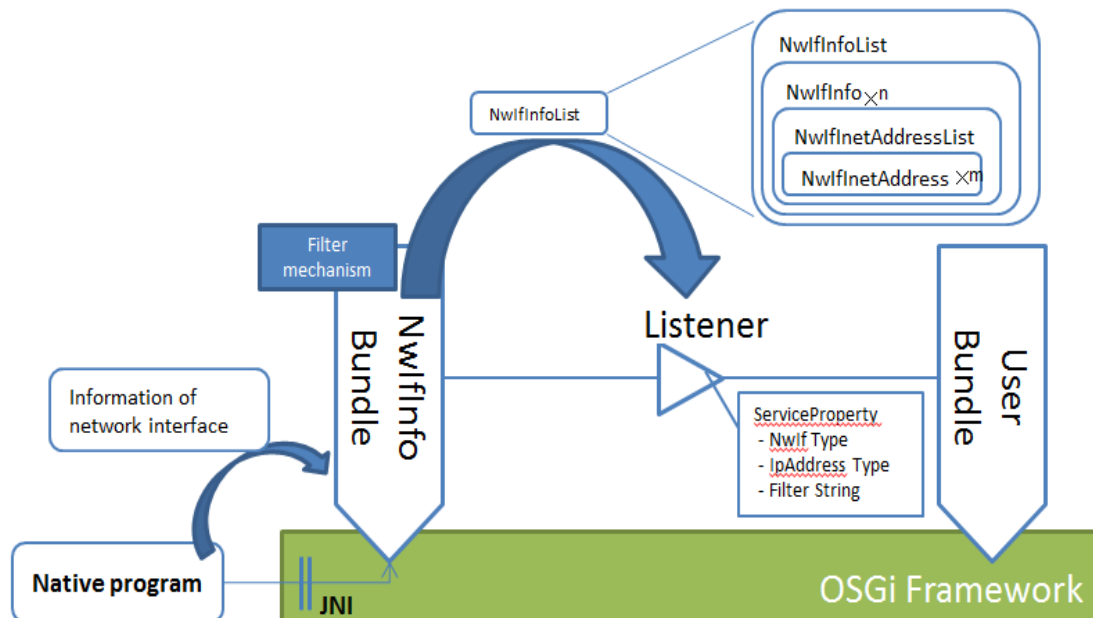
Fig. 3 Overview of Network Interface Information Service

(Whiteboard pattern model)

9   Security Considerations

The user bundles which want to know information of Network Interface should be assigned ServicePermission[NetworkAdapter, GET] and ServicePermission[NetworkAddress, GET].

Filter Based Permissions can also be utilized for assigning ServicePermission. If the platform provider wants to control a bundle to access only the service, the following ServicePermission can be set as an example.

ServicePermission["&(objectClass=org.osgi.service.networkadapter.NetworkAdapter) (service.networkAdapter.type=LAN)", GET]

ServicePermission["&(objectClass=org.osgi.service.networkadapter.NetworkAddress) (service.networkAdapter.type=LAN)(service.ipAddress.version=IPV4)(service.ipAddress.scope=PRIVATE_USE)", GET]

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