RFC-22 (VEG) Namespace

Confidential, Draft
veg-rfc22_namespace-1_00C

21 Pages

Abstract

This document describes the address spaces and naming model for service gateway communication.

0 Document Information

0.1 Table of Contents

All Page Within This Box

0.2 Status

This document specifies a communications name space for the Open Services Gateway Initiative, and requests discussion and suggestions for improvements. Distribution of this document is unlimited within OSGi.

0.3 Acknowledgement

[Acknowledgements]

0.4 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 [1].

`Source code is shown in this typeface.`

0.5 Revision History

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

Revision	Date	Comments
Initial	4/4/01	Anders Rimén, Gatespace ar@gatespace.com Wim De Munck, ACUNIA wim.demunck@acunia.com
V. 1.00B	18/7/01	Johan Agat, Gatespace AB agat@gatespace.com. Updates according to the comments on the VEG meeting in New York 10-11 July 2001.
V.1.00C	29/10/01	Johan Agat, Gatespace AB agat@gatespace.com. Updated API section to reflect changes and clarifications made on the VEG mailing list and on the VEG meeting in Göteborg 29-30 August 2001.

All Page Within This Box

1 Introduction

Services on a Service Gateway must be able to communicate with external entities. How entities are addressed will vary with the transport mechanism used. This document discusses a common namespace to be used for Service Gateway communication, which is layered above the transport mechanism, creating a transport independent namespace.

2 Motivation and Rationale

A common namespace is needed in order for bundles to know how to address peer entities, regardless of the communicative environment they are running in.

3 Requirements

A common namespace shall meet the following requirements:

- It should support routing and provide a flexible and scalable namespace.
- It should have the ability to address entities within a Service Gateway.
- It should provide the ability to create unique addresses.
- It should be large enough not to pose problems in the future.
- It should be independent of the transport mechanism used.
- There should be well-defined addresses for common services.

4 Definitions

When addressing a particular service running on an OSGi host, the address is divided into three parts; service gateway domain, service gateway host and port.

- **Service Gateway Domain (SGD)**
  A service gateway domain is a collection of service gateway hosts with a common administration and security policy.

- **Service Gateway Host (SGH)**
  A service gateway host is an addressable unit in a service gateway domain. One service gateway host is normally one physical machine, but it is possible for one machine to contain multiple hosts possibly belonging to different SGDs, or for a host to span a cluster of several machines.

- **Service Gateway Port (SGP)**
  The port name is used to identify a service or group of services running on the service gateway host.

- **Federated Address**
  The complete address, consisting of SGD, SGH and SGP.

The namespace is illustrated in Figure 1: Namespace.

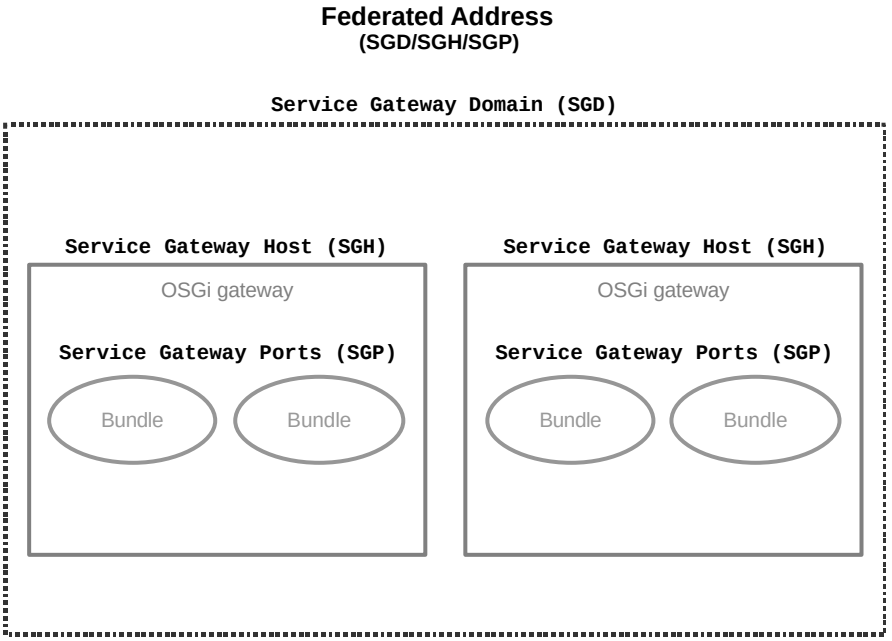

Figure 1: Namespace

All Page Within This Box

4.1 Service Gateway Domain

The domain part of the address specifies what service gateway domain (SGD) a service gateway host belongs to. An OSG operator controls a SGD, and the hosts within the SGD have a common administration, e.g., security and key handling.

The SGD name is a fully qualified domain name as specified in [4], [5] and related documents. The DNS zone with the same name as the SGD MUST be controlled by the OSG operator of the SGD. Two different SGDs MUST NOT have the same name. The SGD MAY be the IP address of the root router of that domain.

```
SGD          ::= FQDN (fully qualified domain name)
```

Equality of SGDs are inherited from equality of FQDNs, i.e., no significance is attached to the case.

4.2 Service Gateway Host

The service gateway host name is an UTF-8 encoded string that uniquely defines a unicast, multicast, anycast or broadcast address within an SGD. It is the responsibility of the OSG operator of the SGD to ensure that service gateway host names are unique within the SGD.

```
SGH          ::= host_name
host_name    ::= UTF-8 encoded string, all characters allowed
```

The OSG operator for the current SGD chooses how the SGH address is designed. This means that two SGH addresses belonging to different SGDs may have different structure.

Note: Since the name `"localhost"` has a special meaning in a Federated Address, see Section 4.4, it SHOULD NOT be chosen as a SGH.

Example: SGD_1 [`sgd1.foo1.com`].

The OSG operator `foo1` has chosen the following SGH structure.

```
SGH : <gateway>.<area>
```

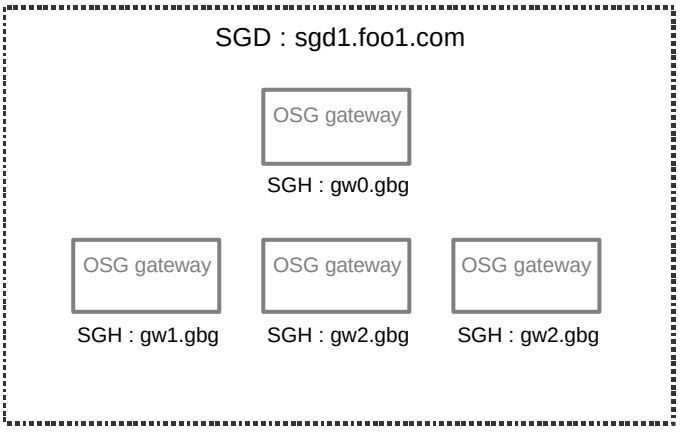

Figure 2: Example 1 SGH

All Page Within This Box

Example: SGD$_2$ [sgd1.foo2.com].

The OSG operator foo2 has chosen the following SGH structure.

```
SGH : <country-code>.<area-code>.<number>
```

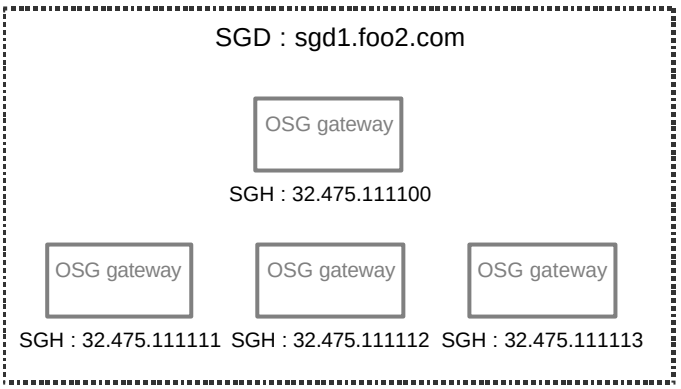

Figure 3: Example 2 SGH

The fact that the structure of two SGHs may differ if they belong to different SGDs implies that a service gateway MUST treat the SGH belonging to another domain as an opaque UTF-8 encoded string.

A service gateway MUST be able to parse and route messages within its domain. This means that the routing algorithm used for host based routing may differ between domains. This RFC suggests that communicative services using host based routing shall allow the host based routing algorithm to be changed by OSG operators.

DNS NAPTR records may exist for a SGD and can be used to translate host names to transport addresses.

Example: NAPTR records for service gateway domain sgd1.foo.com.

```
gw1.gbg     ->        http://gw1.gbg.sgd1.foo.com
            or        sip:gw1.gbg@sgd1.foo.com
```

The details and use of DNS NAPTR records is outside the scope of this document, and shall be described separately.

4.3 Service Gateway Port

The port name identifies a service or a group of services and is expressed as a UTF-8 encoded string. Ports starting with a dollar sign are registered with OSGi. A protocol MUST be specified for registered ports. Ports not starting with a dollar sign may be freely used.

```
SGP          ::= ( unreg_port | reg_port )
unreg_port   ::= port_name
reg_port     ::= '$' port_name
port_name    ::= UTF-8 encoded string, not '$','/'
```

Note: The procedures for registering ports with OSGi are not defined in this document.

All Page Within This Box

Note: Ports used by different communication services are not distinguished. If a given port is in use by, e.g., the ConnectionService, the same port MAY also be available for other communication services, such as the MessageService.

Example: A music distribution application

A music server bundle is installed on a service gateway (SGH : `server.gbg`). A client music bundle is installed on another service gateway (SGH : `gw1.gbg`).

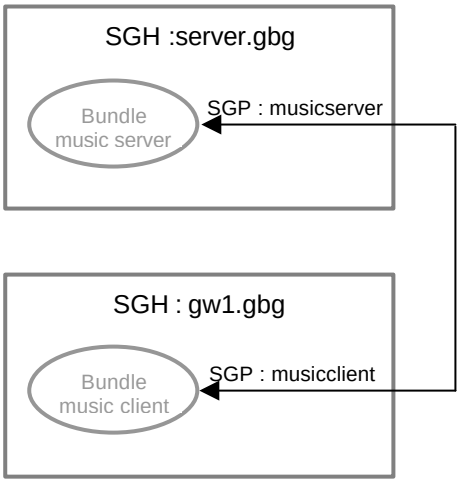

Figure 4: Example SGP

The port used by the music server is `musicserver`, whereas the music client uses the port `musicclient`. These ports are not registered at OSGi, and therefore do not contain any leading '$'. The client requests a tune by sending a message to port `musicserver` on `server.gbg`. The source port is set to `musicclient` and source host to `gw1.gbg`. The music server on the receiving host listening to port `musicserver` receives and processes the message, replying with a new message from port `musicserver` to port `musicclient`. The music client that sent the original message is listening to port `musicclient` and receives the reply.

4.4 Federated Address

A federated address is the complete address, consisting of SGD, SGH and SGP.

```
FEDERATE_ADDRESS ::= [SGD] '/' SGH '/' [SGP]
```

If the SGD part is not specified in a federated address, or if its value is `"localdomain"`, it means that the address applies to the current domain.

If the SGH part has the value `"localhost"`, and the SGD part applies to the current domain, the address applies to one of the addresses assigned to the current service gateway.

If SGP is not specified, it addresses the actual service gateway. This can be useful if certain control messages are used, e.g. host-based key exchange, transmission control messages etc.

Example: Federated addresses

All Page Within This Box

```
sgd1.foo1.com/server.gbg/musicserver
sgd1.foo1.com/gw1.gbg/musicclient
/server.gbg/musicserver
sgd1.foo2.com/32.475.111111/
/32.475.111112/
```

When federated addresses are presented in textual form, they shall follow the above notation. Escape sequences, such as \uXX, poses a problem when presenting addresses for a user. In order for users to be able to distinguish between the character '\uXX' and the string "\uXX" all '\'-characters shall be escaped with an additional '\'. E.g. character '\uXX' is presented as "\uXX", while the string "\uXX" is presented as "\\uXX".

Note: Since the SGH can contain all characters, including `'/'`, parsing a Federated Address needs to be done from both ends.

5 Examples

5.1 Example 1

The company **foo.com** owns and operates a set of service gateways. Some of these gateways reside in homes, whereas the other gateways reside in cars. The example is illustrated in Figure 5: Example overview.

The home gateways have IP-connectivity to foo's backend router and are assigned static IP-addresses.

The mobile gateways are reached using SMS. A backend SMS server handles all SMS traffic to the mobile gateways. The SMS server is also connected to foo's network, having IP-connectivity with the home router.

A database is kept and updated by foo, containing tables mapping SGH to transport addresses.

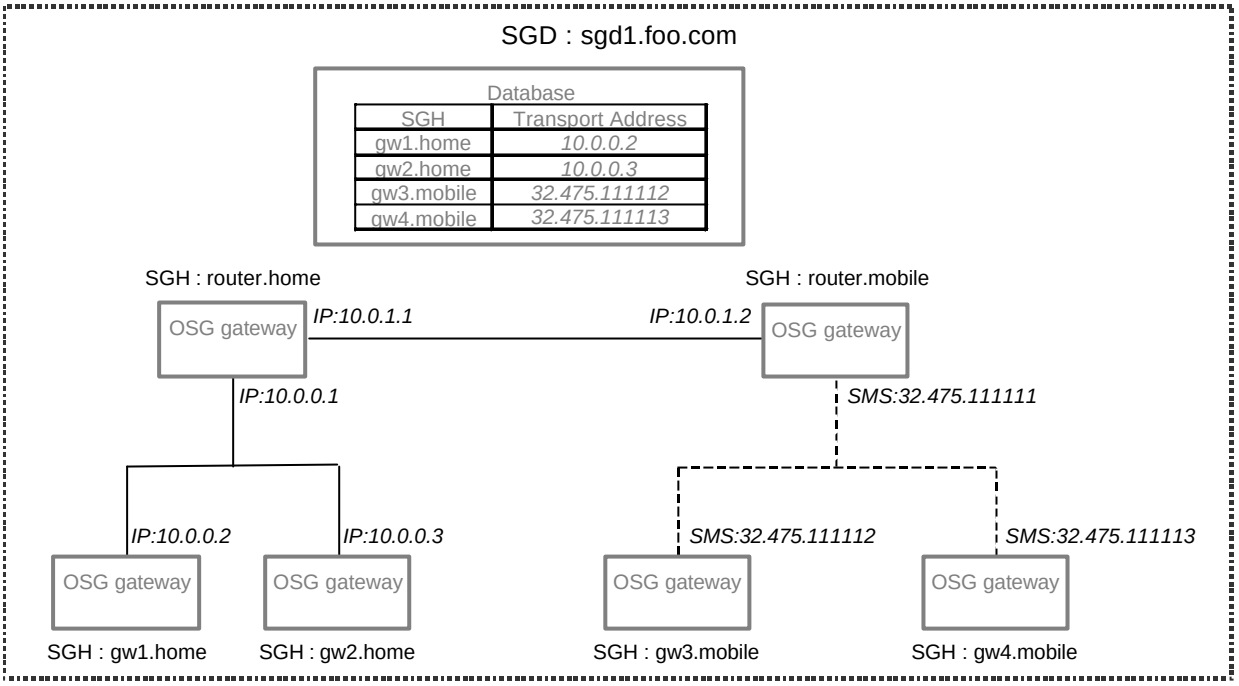

Figure 5: Example overview

5.1.1 Service Gateway Domain

The SGD name for this domain is chosen to be "`sgd1.foo.com`" Since the company owns the DNS zone `foo.com` this is a valid name, and adding `sgd1` to the DNS zone allows for other domains to be created later.

5.1.2 Service Gateway Host Names

The service gateway host name is chosen to be `<gateway>.<area>` where `<gateway>` is the name of a particular gateway and `<area>` specifies whether the gateway reside in a home or a car. In order to simplify mapping to DNS, all area- and gateway names only contain the characters `a-z` and `0-9` and do not start with a digit.

The route tables are designed so that the home gateways route all their traffic to the home router, while the mobile gateways route all their traffic to the mobile router.

Route table: Home gateways

SGH	Transport address
*	*IP:10.0.0.1*

Route table: Mobile gateways

SGH	Transport address
*	*SMS:32.475.111111*

The backend routers are configured so that the home router handles home gateways and the mobile router handles the mobile gateways. They are also configured to route traffic between the IP and SMS network.

All Page Within This Box

Route table: Home router

SGH	Transport address
*.home	Check database
*.mobile	*IP:10.0.1.2*

Route table: Mobile router

SGH	Transport address
*.mobile	Check database
*.home	*IP:10.0.1.1*

DNS NAPTR records could have been used instead of the database.

5.1.3 Service Gateway Ports

The cars have an application installed, HomeController, which allows it to control certain devices in the homes. The devices in the home are either X10 or Jini enabled. Two bundles running on the home gateways, X10Controller and JiniController, control these two groups of devices. These bundles act as proxies, receiving messages, checking security issues (authorization etc.), and possibly talk to the devices in the appropriate protocol (X10 or Jini). The SGP used by these are bundles are `X10Controller` and `JiniController`. The example application is illustrated in Figure 6: Example application.

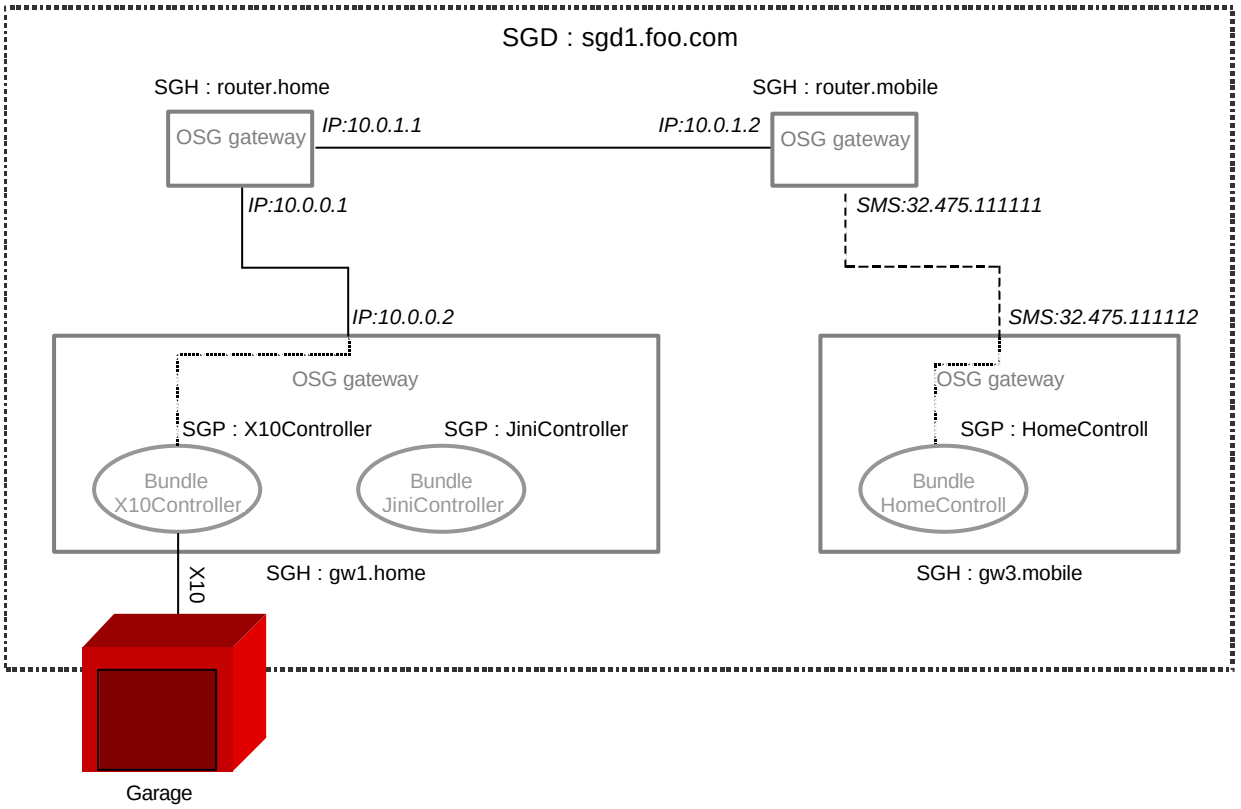

Figure 6: Example application

All Page Within This Box

If a user sitting in the car `gw3.mobile`, wants to open the garage (X10 devices) in his house, `gw1.home`, he will issue this request through the HomeController, which will create a message according to the X10Controller protocol.

Example: X10Control message 'open garage'

```
Source:         sgd1.foo.com/gw3.mobile/HomeControll
Destination:    sgd1.foo.com/gw1.home/X10Controller
Data:           Open garage
```

When the X10Controller receives the message it will, after checking message authentication and authorization, issue a X10 command instructing the garage to open. The result of the command is sent back as an acknowledgement to the mobile gateway.

Example: X10Control message 'open garage ack'

```
Source:         sgd1.foo.com/gw1.home/X10Controll
Destination:    sgd1.foo.com/gw3.mobile/HomeControll
Data:           Garage opened
```

6 API Specification

6.1 org.osgi.util.address
Class FederatedAddress

```
java.lang.Object
  |
  +--org.osgi.util.address.FederatedAddress
```

public class **FederatedAddress**
extends java.lang.Object

A Federated Address is the combination of a Service Gateway Domain, Service Gateway Host and Service Gateway Port.

Service Gateway Domain (SGD)
: A service gateway domain is a collection of service gateway hosts with a common administration and security policy.

Service Gateway Host (SGH)
: A service gateway host is an addressable unit in a service gateway domain. One service gateway host is normally one physical machine, but it is possible for one machine to contain multiple hosts possibly belonging to different SGDs, or for a host to span a cluster of several machines.

Service Gateway Port (SGP)
: The port name is used to identify a service or group of services running on the service gateway host.

All Page Within This Box

Constructor Summary

FederatedAddress(java.lang.String address) Creates a FederatedAddress from a string representation.
FederatedAddress(java.lang.String domain, java.lang.String host, java.lang.String port) Creates a FederatedAddress from three separate strings.

Method Summary

boolean	**equals**(java.lang.Object obj) Determines if this FederatedAddress is equal to the specified object.
java.lang.String	**getDomain**() Returns the Service Gateway Domain (SGD) part of this address.
java.lang.String	**getHost**() Returns the Service Gateway Host (SGH) part of this address.
java.lang.String	**getPort**() Returns the Service Gateway Port (SGP) part of this address.
java.lang.String	**toString**() Returns a string representation of this FederatedAddress object.

Methods inherited from class java.lang.Object
clone, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

6.1.1 FederatedAddress

`public FederatedAddress(java.lang.String address)`

Creates a FederatedAddress from a string representation.

The address must be on the form domain/host/port. The domain localdomain and the host localhost can be used to denote the current gateway.

The domain must be a fully qualified domain name (FQDN), registerd in DNS. The empty string and the string localdomain are also allowed and denote the domain of the current gateway.

An arbitrary string may be used as the host. The string localhost may be used to indicate the host name of the current gateway.

The port may not contain the character '/'. The character '$' must not occur embedded in the port name but may occur as the initial character to indicate a registered port. An empty string can be used to leave the port part of the constructed FederatedAddress unspecified.

Parameters:

address - The fully qualified address, containing Gateway Domain, Gateway Host and Gateway Port, following the syntax described in RFC22.

Throws:

java.lang.IllegalArgumentException - if the address is malformed. No check is performed that the domain is a true FQDN or that it is registered in DNS.

All Page Within This Box

6.1.2 FederatedAddress

```
public FederatedAddress(java.lang.String domain,
                        java.lang.String host,
                        java.lang.String port)
```

Creates a FederatedAddress from three separate strings.

Parameters:

domain - The Gateway domain. Must be a fully qualified domain name (FQDN), registered in DNS. The empty string, null, and the string localdomain are also allowed and denote the domain of the current gateway.

host - The Gateway host. An arbitrary string can be used. The string localhost may be used to indicate the host name of the current gateway.

port - The Gateway port. May not contain the character '/'. The character '$' must not occur embedded in the port name but may occur as the initial character to indicate a registered port. To leave the port part of the FederatedAddress unspecified, null or the empty string can be passed.

Throws:

java.lang.IllegalArgumentException - if the address is malformed. No check is performed that the domain is a true FQDN or that it is registered in DNS.

Method Detail

6.1.3 getDomain

```
public java.lang.String getDomain()
```

Returns the Service Gateway Domain (SGD) part of this address.

Returns:

A fully qualified domain name, controlled by the operator of this domain. The domain is returned in its canoical lowercase form.

If null is returned, this address applies to the same domain as the caller. For FederatedAddresses created with a domain set to localdomain, the empty string or null, getDomain() will return null.

6.1.4 getHost

```
public java.lang.String getHost()
```

Returns the Service Gateway Host (SGH) part of this address. A Service Gateway Host is unique within a Service Gateway Domain. The result of this method is only meaningful within the context of the Service Gateway Domain of this Federated Address.

Returns:

A UTF-8 encoded string that uniquely defines a unicast, mulitcast, anycast or broadcast address within a Service Gateway Domain.

6.1.5 getPort

```
public java.lang.String getPort()
```

Returns the Service Gateway Port (SGP) part of this address. This can be a service or a group of services running on a Service Gateway Host. If the SGP part of this FederatedAddress is not specified, null will be returned.

Returns:

A port identifier.

6.1.6 equals

```
public boolean equals(java.lang.Object obj)
```

Determines if this FederatedAddress is equal to the specified object.

Overrides:

All Page Within This Box

equals in class java.lang.Object

Returns:

true iff this FederatedAddress is equal to the specified FederatedAddress object. Two FederatedAddress objects are considered equal if they have identical Host and Port fields and their Domain fields are equal ignoring case.

The domain name localdomain is considered equal to the empty domain name and a domain specified with null. However, the FederatedAddress class is not aware of the real domain of a gateway and thus cannot equate localdomain with that.

6.1.7 toString

`public java.lang.String` **`toString`**`()`

Returns a string representation of this FederatedAddress object. The string returned is a valid string representation of a FederatedAddress, as accepted by FederatedAddress(String).

Overrides:

toString in class java.lang.Object

Returns:

A string representing this FederatedAddress, encoded on the form domain_name/host_name/port_name.

6.2 org.osgi.util.address
Class FederatedAddressPermission

```
java.lang.Object
  |
  +--java.security.Permission
        |
        +--org.osgi.util.address.FederatedAddressPermission
```

All Implemented Interfaces:

java.security.Guard, java.io.Serializable

---

public final class **FederatedAddressPermission**
extends java.security.Permission

FederatedAddressPermission specifies a bundle's right to use the FederatedAddress name space. It is used by the MessageService and the ConnectionService to provide fine-grained control over the utilization of those services.

The actions to be granted are passed to the constructor in a string containing a list of zero or more comma-separated keywords. The allowed keywords are listen, sendTo, sendFrom, accept and connect. Their semantics is defined as follows:

action: listen

Permission to listen for messages on the addresses matching the name argument specified in the constructor.

action: sendFrom

Permission to send messages from the addresses matching the name argument specified in the constructor.

action: sendTo

Permission to send messages to the addresses matching the name argument specified in the constructor.

action: accept

Permission to accept connections on the addresses matching the name argument specified in the constructor.

action: connect

Permission to create connections to the addresses matching the name argument specified in the constructor.

All Page Within This Box

Case is not significant when actions are specified.
The permission name specifies the federated address(es) that this permission applies to. The name consists of three parts separated by '/', i.e., domain_name/host_name/port_name. Each part may contain at most one wildcard asterisk, '*'. In the domain part, the asterisk must occur alone or in the leftmost position followed by a dot '.'. In the host part, the asterisk may occur by itself or embedded anywhere in the host string. In the port part the asterisk must occur alone or in the rightmost position.
If the character '*' is part of the host_name or port_name as a normal character and not a wildcard, it must be escaped with a preceding '\'. Furthermore, the character '\' is itself escaped with a '\' in the standard way.

Example:

The example illustrates the permission needed for a payment system, consisting of a payment server and payment clients that use the MessageService to communicate. The payment server receives requests from payment clients running on foreign service gateways. The server and clients all run on service gateways residing in the same service gateway domain, 'sgd.foo.com'. The payment server runs on the service gateway host 'server', using the port 'pay'. The clients all use the port 'payclient' as the reply port when they contact the server.
The clients all need permission to sent to the servers 'pay' port and to specify their own 'payclient' port as the reply port when they send. Naturally, they also need permission to listen for incoming messages on the port 'payclient'.

```
  FederatedAddressPermission permClient1 =
   new FederatedAddressPermission("sgd.foo.com/server/pay", "sendTo");
  FederatedAddressPermission permClient2 =
   new FederatedAddressPermission("sgd.foo.com/localhost/payclient", "listen,
sendFrom");
```

The server needs permission to listen on and send from the port 'pay' and to send to all clients 'payclient' port.

```
  FederatedAddressPermission permServer1 =
   new FederatedAddressPermission("sgd.foo.com/server/pay" "listen, sendFrom");
  FederatedAddressPermission permServer2 =
   new FederatedAddressPermission("sgd.foo.com/*/payclient;", "sendTo");
```

See Also:

Serialized Form

Field Summary

static java.lang.String	**ACCEPT** Accept action.
static java.lang.String	**CONNECT** Connect action.
static java.lang.String	**LISTEN** Listen action.
static java.lang.String	**SEND_FROM** SendFrom action.
static java.lang.String	**SEND_TO** SendTo action.

All Page Within This Box

Constructor Summary

FederatedAddressPermission(java.lang.String name, java.lang.String actions) Creates a FederatedAddressPermission object.	

Method Summary

boolean	**equals**(java.lang.Object obj) Checks two FederatedAddressPermission objects for equality.
java.lang.String	**getActions**() Returns the canonical string representation of the actions, separated by comma, in alphabetical order.
int	**hashCode**() Returns the hash code of this FederatedAddressPermission, which is computed by appending the name of this FederatedAddressPermission to the result of getActions() and computing the hash code of the resulting string.
boolean	**implies**(java.security.Permission p) Checks if this FederatedAddressPermission object "implies" the specified permission *p*.
java.security.PermissionCollection	**newPermissionCollection**() Returns a new PermissionCollection object for storing FederatedAddressPermission objects.

Methods inherited from class java.security.Permission
checkGuard, getName, toString

Methods inherited from class java.lang.Object
clone, finalize, getClass, notify, notifyAll, wait, wait, wait

Field Detail

6.2.1 LISTEN

```
public static final java.lang.String LISTEN
```

Listen action. Used by MessageService.

6.2.2 SEND_TO

```
public static final java.lang.String SEND_TO
```

SendTo action. Used by MessageService.

6.2.3 SEND_FROM

```
public static final java.lang.String SEND_FROM
```

SendFrom action. Used by MessageService.

All Page Within This Box

6.2.4 ACCEPT

`public static final java.lang.String` **`ACCEPT`**

Accept action. Used by ConnectionService.

6.2.5 CONNECT

`public static final java.lang.String` **`CONNECT`**

Connect action. Used by ConnectionService.

Constructor Detail

6.2.6 FederatedAddressPermission

```
public FederatedAddressPermission(java.lang.String name,
                                  java.lang.String actions)
```

Creates a FederatedAddressPermission object.

Parameters:

name - The name of the addresses credential or property that needs to be accessed. It should be on the form "domain/host/port/, where each part may contain one or zero wildcards '*'.

action - Should be a comma separated list containing one or more of LISTEN, SEND_TO, SEND_FROM, ACCEPT or SEND.

Throws:

java.lang.IllegalArgumentException - if the name or actions cannot be parsed.

Method Detail

6.2.7 equals

`public boolean` **`equals`**`(java.lang.Object obj)`

Checks two FederatedAddressPermission objects for equality. Checks that obj is a FederatedAddressPermission, and has the same name and actions as this object.

The domain name localdomain is considered equal to the empty domain name. However, the FederatedAddressPermission class is not aware of the real domain of a gateway and thus cannot equate localdomain with that.

Overrides:

equals in class java.security.Permission

Parameters:

obj - the object to be compared for equality with this object.

Returns:

true if obj is a FederatedAddressPermission, and has the same name and actions as this FederatedAddressPermission object.

6.2.8 getActions

`public java.lang.String` **`getActions`**`()`

Returns the canonical string representation of the actions, separated by comma, in alphabetical order.

Overrides:

getActions in class java.security.Permission

Returns:

the canonical string representation of the actions.

All Page Within This Box

6.2.9 hashCode

`public int` **`hashCode`**`()`

Returns the hash code of this FederatedAddressPermission, which is computed by appending the name of this FederatedAddressPermission to the result of getActions() and computing the hash code of the resulting string.

Overrides:

hashCode in class java.security.Permission

6.2.10 implies

`public boolean` **`implies`**`(java.security.Permission p)`

Checks if this FederatedAddressPermission object "implies" the specified permission *p*.

More specifically, this method returns true iff:

- *p* is an instanceof FederatedAddressPermission,
- *p*'s set of actions is a subset of this object's action set, and
- *p*'s name is implied by this object's name.

The domain name localdomain is considered equal to the empty domain name. However, the FederatedAddressPermission class is not aware of the real domain of a gateway and thus cannot equate localdomain with that.

Overrides:

implies in class java.security.Permission

Parameters:

p - The permission to check against.

Returns:

true if the specified permission is implied by this object; false otherwise.

6.2.11 newPermissionCollection

`public java.security.PermissionCollection` **`newPermissionCollection`**`()`

Returns a new PermissionCollection object for storing FederatedAddressPermission objects.

Overrides:

newPermissionCollection in class java.security.Permission

Returns:

a new PermissionCollection object suitable for storing FederatedAddressPermission objects.

All Page Within This Box

7 Security Considerations

No special security issues are foreseen for this definition and API.

8 Document Support

8.1 References

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

[2]. IETF, The Naming Authority Pointer (NAPTR) DNS Resource Record, RFC 2915, September 2000.

[3]. IETF, Uniform Resource Identifiers (URI): Generic Syntax, RFC 2396, August 1998.

[4]. IETF, Domain names - concepts and facilities, RFC 1034, November 1987.

[5]. IETF, Domain names - implementation and specification, RFC 1035, November 1987.

8.2 Author's Address

Name	Johan Agat
Company	Gatespace
Address	
Voice	
e-mail	agat@gatespace.com

Name	Anders Rimén
Company	Gatespace
Address	
Voice	

All Page Within This Box

e-mail	ar@gatespace.com

Name	Andreas Gunnarsson
Company	Gatespace
Address	
Voice	
e-mail	andreas@gatespace.com

Name	Wim De Munck
Company	ACUNIA
Address	
Voice	
e-mail	Wim.demunck@acunia.com

Name	Jean-Philippe Cattoor
Company	ACUNIA
Address	
Voice	
e-mail	cattoor@gatespace.com

8.3 Acronyms and Abbreviations

8.4 End of Document

All Page Within This Box