RFC 10 – User Management API

Members Only, Final
cpeg-rfc_10_usermanagement-1_00

25 Pages

Abstract

The UserAdmin and related interfaces presented in this RFC allow a gateway administrator to configure a gateway with users who should be granted access to the gateway. Each user can be configured with properties, credentials, and roles. The user information maintained by the UserAdmin determines which users should be granted access to the gateway, and which privileges (if any) they should be given. User credentials are used to authenticate incoming requests into the gateway. Once a request has been authenticated, authorization information corresponding to the remote user can be determined and used to control access to protected resources and operations in the gateway.

1 Document Information

1.1 Table of Contents

All Page Within This Box

1.2 Status

This document specifies the UserAdmin and related interfaces for the Open Services Gateway Initiative, and requests discussion and suggestions for improvements. Distribution of this document is unlimited within OSGi.

1.3 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.`

1.4 Revision History

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

All Page Within This Box

Revision	Date	Comments
First draft	January 12, 2001	First draft in new OSGi RFC format. As agreed at the Miami meeting, the following changes have been made: 1. Renamed `Entity` to `User`, and specified what `User` means (i.e., more than just a human being). 2. Renamed `Authorization.hasGroup` to `Authorization.isMember`. 3. Removed `Dictionary` argument from `Entity.getAuthorization`.
Second draft	February 16, 2001	1. Introduced a role concept, using the base interface `Role`. `User` extends `Role` and `Group` extends `User`. 2. Added `UserAdminListeners`, which are called when the `UserAdmin` database is modified.
Third draft	March 29, 2001	1. Added clarifying examples explaining permissions, security considerations and group semantics. 2. Introduced the pre-defined role “user.anyone”.
Final	April 12, 2001	1. Added clarifying examples explaining permissions, security considerations and group semantics. 2. Introduced the pre-defined role “user.anyone”.

2 Introduction

This interface allows gateway entry points to authenticate incoming requests and derive authorization information about the requestor. This information can then be passed to bundle code responsible for servicing the request. Based on this information, the code servicing the request will either grant or deny access to protected resources.

A gateway administrator uses this service to define gateway users and configure them with properties, credentials, and roles. A gateway may have several entry points, each of which will be responsible for authenticating incoming requests. One example of a gateway entry point is the HttpService, which delegates authentication of incoming requests to the handleSecurity method of the HttpContext that was specified when the servlet or resource that is the target of the request was registered.

All Page Within This Box

The gateway entry points will use the information configured in the UserAdmin to authenticate incoming requests. Once a request has been authenticated, the authorization information corresponding to the (remote) requestor is determined. Authorization is expressed in terms of the authenticated user possessing certain predefined roles. Bundle code responsible for servicing a request will either deny or grant access to protected resources and operations based on which roles the requestor possesses.

3 Motivation and Rationale

The OSGi SG 1.0 specification provides the means of access controls based on where bundle code came from and who signed it. The need for such access controls derives from the ability of service gateways to download bundles from the network and run them locally. This specification, however, lacks the means to enforce similar access controls based on who runs the bundle code. To provide this type of access control, the OSGi SG specification requires additional support for authentication (determining who's actually running the bundle code) and authorization.

4 API Specification

4.1 org.osgi.service.useradmin Interface UserAdmin

public interface **UserAdmin**

This interface is used to manage a database of named roles, which can be used for authentication and authorization purposes.

This version of UserAdmin defines two types of roles: "User" and "Group". Each type of role is represented by an "int" constant and an interface. The range of positive integers is reserved for new types of roles that may be added in the future. When defining proprietary role types, negative constant values must be used.

Every role has a name and a type.

A `User` role can be configured with credentials (e.g., a password) and properties (e.g., a street address, phone number, etc.).

All Page Within This Box

A Group role represents an aggregation of User and Group roles. In other words, the members of a Group role are roles themselves.

Every UserAdmin manages and maintains its own namespace of roles, in which each role has a unique name.

Method Summary	
Role	**createRole**(java.lang.String name, int type) Creates a role with the given name and of the given type.
Authorization	**getAuthorization**(User user) Creates an Authorization object that encapsulates the specified user and the roles it possesses.
Role	**getRole**(java.lang.String name) Gets the role with the given name from this UserAdmin.
Role[]	**getRoles**(java.lang.String filter) Gets the roles managed by this UserAdmin that have properties matching the specified LDAP filter criteria.
User	**getUser**(java.lang.String key, java.lang.String value) Gets the user with the given property key-value pair from the UserAdmin database.
boolean	**removeRole**(java.lang.String name) Removes the role with the given name from this UserAdmin.

Method Detail

4.1.1 createRole

```
public Role createRole(java.lang.String name,
                       int type)
```

Creates a role with the given name and of the given type.

If a role was created, a UserAdminEvent of type UserAdminEvent.ROLE_CREATED is broadcast to any UserAdminListener.

Parameters:
name - The name of the role to create.
type - The type of the role to create. Must be either Role.USER or Role.GROUP.
Returns:
The newly created role, or null if a role with the given name already exists.
Throws:
java.lang.IllegalArgumentException - if type is invalid.
java.lang.SecurityException - If a security manager exists and the caller does not have the UserAdminPermission with name admin.

4.1.2 removeRole

```
public boolean removeRole(java.lang.String name)
```

Removes the role with the given name from this UserAdmin.

All Page Within This Box

If the role was removed, a UserAdminEvent of type `UserAdminEvent.ROLE_REMOVED` is broadcast to any UserAdminListener.

Parameters:
`name` - The name of the role to remove.
Returns:
`true` If a role with the given name is present in this UserAdmin and could be removed, otherwise `false`.
Throws:
`java.lang.SecurityException` - If a security manager exists and the caller does not have the `UserAdminPermission` with name `admin`.

4.1.3 getRole

```
public Role getRole(java.lang.String name)
```

Gets the role with the given name from this UserAdmin.
Parameters:
`name` - The name of the role to get.
Returns:
The requested role, or `null` if this UserAdmin does not have a role with the given name.

4.1.4 getRoles

```
public Role[] getRoles(java.lang.String filter)
                throws org.osgi.framework.InvalidSyntaxException
```

Gets the roles managed by this UserAdmin that have properties matching the specified LDAP filter criteria. See `org.osgi.framework.Filter` or IETF RFC 2254 for a description of the filter syntax. If a `null` filter is specified, all roles managed by this UserAdmin are returned.
Parameters:
`filter` - The filter criteria to match.
Returns:
The roles managed by this UserAdmin whose properties match the specified filter criteria, or all roles if a `null` filter is specified.

4.1.5 getUser

```
public User getUser(java.lang.String key,
                    java.lang.String value)
```

Gets the user with the given property key-value pair from the UserAdmin database. This is a convenience method for retrieving a user based on a property for which every user is supposed to have a unique value (within the scope of this UserAdmin), such as a user's X.500 distinguished name.
Parameters:
`key` - The property key to look for.
`value` - The property value to compare with.
Returns:
A matching user, if *exactly* one is found. If zero or more than one matching users are found, `null` is returned.

4.1.6 getAuthorization

```
public Authorization getAuthorization(User user)
```

All Page Within This Box

Creates an Authorization object that encapsulates the specified user and the roles it possesses. The `null` user is interpreted as the anonymous user.

Parameters:

`user` - The user to create an Authorization object for, or `null` for the anonymous user.

Returns:

the Authorization object for the specified user.

4.2 org.osgi.service.useradmin Interface Role

All Known Subinterfaces:

Group, User

public interface **Role**

The base interface for Role objects managed by the `UserAdmin` service.

This interface exposes the characteristics shared by all Roles: a name, a type, and a set of properties.

Properties represent public information about the Role that can be read by anyone. Specific `UserAdminPermission`s are required to change a Role's properties.

Role properties are Dictionary objects. Changes to these objects are propagated to the `UserAdmin` service and made persistent.

Every UserAdmin contains a set of predefined roles that are always present and cannot be removed. All predefined roles are of type `ROLE`. This version of the `org.osgi.service.useradmin` package defines a single predefined role named "user.anyone", which is inherited by any other role. Other predefined roles may be added in the future.

Field Summary	
`static int`	**`GROUP`** The type of a `Group` role.
`static int`	**`ROLE`** The type of a predefined role.
`static int`	**`USER`** The type of a `User` role.

Method Summary	
`java.lang.String`	**`getName`**`()` Returns the name of this role.
`java.util.Dictionary`	**`getProperties`**`()` Returns a Dictionary of the (public) properties of this Role.
`int`	**`getType`**`()` Returns the type of this role.

All Page Within This Box

Field Detail

4.2.1 ROLE

`public static final int` **`ROLE`**

The type of a predefined role.

The value of `ROLE` is 0.

4.2.2 USER

`public static final int` **`USER`**

The type of a `User` role.

The value of `USER` is 1.

4.2.3 GROUP

`public static final int` **`GROUP`**

The type of a `Group` role.

The value of `GROUP` is 2.

Method Detail

4.2.4 getName

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

Returns the name of this role.

Returns:

The role's name.

4.2.5 getType

`public int` **`getType`**`()`

Returns the type of this role.

Returns:

The role's type.

4.2.6 getProperties

`public java.util.Dictionary` **`getProperties`**`()`

Returns a Dictionary of the (public) properties of this Role. Any changes to the returned Dictionary will change the properties of this Role. This will cause a UserAdminEvent of type `UserAdminEvent.ROLE_CHANGED` to be broadcast to any UserAdminListeners.

Only objects of type `String` may be used as property keys, and only objects of type `String` or `byte[]` may be used as property values. Any other types will cause an exception of type `IllegalArgumentException` to be raised.

All Page Within This Box

In order to add, change, or remove a property in the returned Dictionary, a `UserAdminPermission` named after the property name (or a prefix of it) with action `changeProperty` is required.

Returns:
Dictionary containing the properties of this Role.

4.3 org.osgi.service.useradmin
Interface User

All Superinterfaces:
Role
All Known Subinterfaces:
Group

public interface **User**
extends Role

A User managed by a `UserAdmin` service.

In this context, the term "user" is not limited to just human beings. Instead, it refers to any entity that may have any number of credentials associated with it that it may use to authenticate itself.

In general, User objects are associated with a specific `UserAdmin` service (namely the one that created them), and cannot be used with other UserAdmin services.

A User may have credentials (and properties, inherited from `Role`) associated with it. Specific `UserAdminPermission`s are required to read or change a User's credentials.

Credentials are Dictionary objects and have semantics that are similar to the properties in Role.

Fields inherited from interface org.osgi.service.useradmin.Role
`GROUP`, `ROLE`, `USER`

Method Summary	
`java.util.Dictionary`	**`getCredentials`**`()` Returns a Dictionary of the credentials of this User.
`boolean`	**`hasCredential`**`(java.lang.String key, java.lang.Object value)` Checks to see if this User has a credential with the specified key set to the specified value.

Methods inherited from interface org.osgi.service.useradmin.Role
`getName`, `getProperties`, `getType`

Method Detail

All Page Within This Box

4.3.1 getCredentials

`public java.util.Dictionary` **`getCredentials`**`()`

Returns a Dictionary of the credentials of this User. Any changes to the returned Dictionary will change the credentials of this User. This will cause a UserAdminEvent of type `UserAdminEvent.ROLE_CHANGED` to be broadcast to any UserAdminListeners.

Only objects of type String may be used as credential keys, and only objects of type `String` or of type `byte[]` may be used as credential values. Any other types will cause an exception of type `IllegalArgumentException` to be raised.

In order to retrieve a credential from the returned Dictionary, a `UserAdminPermission` named after the credential name (or a prefix of it) with action `getCredential` is required.

In order to add or remove a credential from the returned Dictionary, a `UserAdminPermission` named after the credential name (or a prefix of it) with action `changeCredential` is required.

Returns:
Dictionary containing the credentials of this User.

4.3.2 hasCredential

```
public boolean hasCredential(java.lang.String key,
                             java.lang.Object value)
```

Checks to see if this User has a credential with the specified key set to the specified value.

If the specified credential value is not of type `String` or `byte[]`, it is ignored, that is, `false` is returned (as opposed to an `IllegalArgumentException` being raised).

Parameters:
`key` - The credential key.
`value` - The credential value.
Returns:
`true` if this user has the specified credential; `false` otherwise.
Throws:
`java.lang.SecurityException` - If a security manager exists and the caller does not have the `UserAdminPermission` named after the credential key (or a prefix of it) with action `getCredential`.

4.4 org.osgi.service.useradmin Interface Group

All Superinterfaces:
Role, User

---

public interface **Group**
extends User

A named grouping of roles.

Whether or not a given authorization context implies a Group role depends on the members of that role.

All Page Within This Box

A Group role can have two kinds of member roles: *basic* and *required*. A Group role is implied by an authorization context if all of its required member roles are implied and at least one of its basic member roles is implied.

A Group role must contain at least one basic member role in order to be implied. In other words, a Group without any basic member roles is never implied by any authorization context.

A User role always implies itself.

No loop detection is performed when adding members to groups, which means that it is possible to create circular implications. Loop detection is instead done when roles are checked. The semantics is that if a role depends on itself (i.e., there is an implication loop), the role is not implied.

The rule that a group must have at least one basic member to be implied is motivated by the following example:

```
group foo
  required members: marketing
  basic members: alice, bob
```

Privileged operations that require membership in "foo" can be performed only by alice and bob, who are in marketing.

If alice and bob ever transfer to a different department, anybody in marketing will be able to assume the "foo" role, which certainly must be prevented. Requiring that "foo" (or any Group role for that matter) must have at least one basic member accomplishes that.

However, this would make it impossible for a group to be implied by just its required members. An example where this implication might be useful is the following declaration: "Any citizen who is an adult is allowed to vote." An intuitive configuration of "voter" would be:

```
group voter
  required members: citizen, adult
     basic members:
```

However, according to the above rule, the "voter" role could never be assumed by anybody, since it lacks any basic members. In order to address this deficiency a predefined role named "user.anyone" can be specified, which is always implied. The desired implication of the "voter" group can then be achieved by specifying "user.anyone" as its basic member, as follows:

```
group voter
  required members: citizen, adult
     basic members: user.anyone
```

Fields inherited from interface org.osgi.service.useradmin.Role
GROUP, ROLE, USER

Method Summary

All Page Within This Box

boolean	**addMember**(Role role) Adds the specified role as a basic member to this Group.
boolean	**addRequiredMember**(Role role) Adds the specified role as a required member to this Group.
Role[]	**getMembers**() Gets the basic members of this Group.
Role[]	**getRequiredMembers**() Gets the required members of this Group.
boolean	**removeMember**(Role role) Removes the specified role from this Group.

Methods inherited from interface org.osgi.service.useradmin.User
getCredentials, hasCredential

Methods inherited from interface org.osgi.service.useradmin.Role
getName, getProperties, getType

Method Detail

4.4.1 addMember

`public boolean` **`addMember`**`(Role role)`

Adds the specified role as a basic member to this Group.

Parameters:

`role` - The role to add as a basic member.

Returns:

`true` if the given role could be added as a basic member, and `false` if this Group already contains a role whose name matches that of the specified role.

Throws:

`java.lang.SecurityException` - If a security manager exists and the caller does not have the `UserAdminPermission` with name `admin`.

4.4.2 addRequiredMember

`public boolean` **`addRequiredMember`**`(Role role)`

Adds the specified role as a required member to this Group.

Parameters:

`role` - The role to add as a required member.

Returns:

`true` if the given role could be added as a required member, and `false` if this Group already contains a role whose name matches that of the specified role.

Throws:

`java.lang.SecurityException` - If a security manager exists and the caller does not have the `UserAdminPermission` with name `admin`.

All Page Within This Box

4.4.3 removeMember

`public boolean` **`removeMember`**`(Role role)`

Removes the specified role from this Group.

Parameters:

`role` - The role to remove from this Group.

Returns:

`true` if the role could be removed, otherwise `false`.

Throws:

`java.lang.SecurityException` - If a security manager exists and the caller does not have the `UserAdminPermission` with name `admin`.

4.4.4 getMembers

`public Role[]` **`getMembers`**`()`

Gets the basic members of this Group.

Returns:

The basic members of this Group, or `null` if this Group does not contain any basic members.

4.4.5 getRequiredMembers

`public Role[]` **`getRequiredMembers`**`()`

Gets the required members of this Group.

Returns:

The required members of this Group, or `null` if this Group does not contain any required members.

4.5 org.osgi.service.useradmin Interface Authorization

public interface **Authorization**

This interface encapsulates an authorization context on which bundles can base authorization decisions where appropriate.

Bundles associate the privilege to access restricted resources or operations with roles. Before granting access to a restricted resource or operation, a bundle will check if the Authorization object passed to it possesses the required role, by calling its hasRole method.

Authorization contexts are instantiated by calling
`UserAdmin.getAuthorization(org.osgi.service.useradmin.User)`

Trusting Authorization objects.

There are no restrictions regarding the creation of Authorization objects. Hence, a service must only accept Authorization objects from bundles that has been authorized to use the service using code based (or Java 2) permissions.

All Page Within This Box

In some cases it is useful to use ServicePermissions to do the code based access control. A service basing user access control on Authorization objects passed to it, will then require that a calling bundle has the ServicePermission to get the service in question. This is the most convenient way. The framework will do the code based permission check when the calling bundle attempts to get the service from the service registry.

Example: A servlet using a service on a user's behalf. The bundle with the servlet must be given the ServicePermission to get the Service.

However, in some cases the code based permission checks need to be more fine-grained. A service might allow all bundles to get it, but require certain code based permissions for some of its methods.

Example: A servlet using a service on a user's behalf, where some service functionality is open to anyone, and some is restricted by code based permissions. When a restricted method is called (e.g., one handing over an Authorization object), the service explicitly checks that the calling bundle has permission to make the call.

Method Summary	
java.lang.String	**getName**() Gets the name of the User that this Authorization context was created for.
java.lang.String[]	**getRoles**() Gets the names of all roles encapsulated by this Authorization context.
boolean	**hasRole**(java.lang.String name) Checks if the role with the specified name is implied by this Authorization context.

Method Detail

4.5.1 getName

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

Gets the name of the User that this Authorization context was created for.

Returns:

The name of the User that this Authorization context was created for, or `null` if no user was specified when this Authorization context was created.

4.5.2 hasRole

`public boolean` **`hasRole`**`(java.lang.String name)`

Checks if the role with the specified name is implied by this Authorization context.

Bundles must define globally unique role names that are associated with the privilege of accessing restricted resources or operations. System administrators will grant users access to these resources, by creating a Group for each role and adding Users to it.

Parameters:

name - The name of the role to check for.

All Page Within This Box

Returns:
`true` if this Authorization context implies the specified role, otherwise `false`.

4.5.3 getRoles

`public java.lang.String[] getRoles()`
Gets the names of all roles encapsulated by this Authorization context.
Returns:
The names of all roles encapsulated by this Authorization context, or `null` if no roles are in the context.

4.6 org.osgi.service.useradmin Class UserAdminEvent

```
java.lang.Object
  |
  +-org.osgi.service.useradmin.UserAdminEvent
```

---

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

Role change event. UserAdminEvents are delivered asynchronously to any UserAdminListeners when a change occurs in any of the Roles managed by a `UserAdmin`.

A type code is used to identify the event. The following event types are defined: `ROLE_CREATED`, `ROLE_CHANGED`, and `ROLE_REMOVED`. Additional event types may be defined in the future.

See Also:
`UserAdmin`, `UserAdminListener`

Field Summary	
`static int`	**`ROLE_CHANGED`** A role has been modified.
`static int`	**`ROLE_CREATED`** A role has been created.
`static int`	**`ROLE_REMOVED`** A role has been removed.

Constructor Summary
`UserAdminEvent``(org.osgi.framework.ServiceReference ref, int type, Role role)` Constructs a UserAdminEvent from the given ServiceReference, event type, and role.

Method Summary	
`Role`	**`getRole`**`()` Gets the `Role` this event was generated for.
`org.osgi.framework.ServiceReference`	**`getServiceReference`**`()` Gets the ServiceReference of the UserAdmin that

All Page Within This Box

	generated this event.
int	**getType**() Returns the type of this event.

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

Field Detail

4.6.1 ROLE_CREATED

```
public static final int ROLE_CREATED
```
A role has been created.

The value of ROLE_CREATED is 0x00000001.

4.6.2 ROLE_CHANGED

```
public static final int ROLE_CHANGED
```
A role has been modified.

The value of ROLE_CHANGED is 0x00000002.

4.6.3 ROLE_REMOVED

```
public static final int ROLE_REMOVED
```
A role has been removed.

The value of ROLE_REMOVED is 0x00000004.

Constructor Detail

4.6.4 UserAdminEvent

```
public UserAdminEvent(org.osgi.framework.ServiceReference ref,
                      int type,
                      Role role)
```
Constructs a UserAdminEvent from the given ServiceReference, event type, and role.

Parameters:

ref - The ServiceReference of the UserAdmin that generated this event.

type - The event type.

role - The role on which this event occurred.

Method Detail

4.6.5 getServiceReference

All Page Within This Box

`public org.osgi.framework.ServiceReference` **`getServiceReference`**`()`

Gets the ServiceReference of the UserAdmin that generated this event.

Returns:

The UserAdmin's ServiceReference.

4.6.6 getType

`public int` **`getType`**`()`

Returns the type of this event. The type values are `ROLE_CREATED`, `ROLE_CHANGED`, and `ROLE_REMOVED`.

Returns:

The event type.

4.6.7 getRole

`public Role` **`getRole`**`()`

Gets the `Role` this event was generated for.

Returns:

The `Role` this event was generated for.

4.7 org.osgi.service.useradmin Interface UserAdminListener

---

public interface **UserAdminListener**

Listener for UserAdminEvents.

UserAdminListeners are registered with the OSGi service registry and notified with a UserAdminEvent when a Role has been created, removed, or modified.

UserAdminListeners can further inspect the received UserAdminEvent to determine its type, the Role it occurred on, and the UserAdmin that generated it.

See Also:

`UserAdmin`, `UserAdminEvent`

Method Summary	
`void`	**`roleChanged`**`(UserAdminEvent event)` Receives notification that a Role has been created, removed, or modified.

Method Detail

4.7.1 roleChanged

`public void` **`roleChanged`**`(UserAdminEvent event)`

Receives notification that a Role has been created, removed, or modified.

Parameters:

`event` - The UserAdminEvent.

All Page Within This Box

4.8 org.osgi.service.useradmin
Class UserAdminPermission

```
java.lang.Object
  |
  +-java.security.Permission
        |
        +-java.security.BasicPermission
              |
              +-org.osgi.service.useradmin.UserAdminPermission
```

All Implemented Interfaces:
java.security.Guard, java.io.Serializable

---

public class **UserAdminPermission**
extends java.security.BasicPermission
implements java.io.Serializable

Permission to configure and access the `Role`s managed by a `UserAdmin`.

This class represents access to the Roles managed by a UserAdmin and their properties and credentials (in the case of `User` roles).

The permission name is the name (or name prefix) of a property or credential. The naming convention follows the hierarchical property naming convention. Also, an asterisk may appear at the end of the name, following a ".", or by itself, to signify a wildcard match. For example: "org.osgi.security.protocol.*" or "*" is valid, but "*protocol" or "a*b" are not valid.

The UserAdminPermission with the reserved name "admin" represents the permission required for creating and removing roles in the UserAdmin, as well as adding and removing members in a Group. This UserAdminPermission does not have any actions associated with it.

The actions to be granted are passed to the constructor in a string containing a list of one or more comma-separated keywords. The possible keywords are: "changeProperty", "changeCredential", and "getCredential". Their meaning is defined as follows:

action: "changeProperty"
: Permission to change (i.e., add and remove) Role properties whose names start with the `name` argument specified in the constructor.

action: "changeCredential"
: Permission to change (i.e., add and remove) User credentials whose names start with the `name` argument specified in the constructor.

action: "getCredential"
: Permission to retrieve and check for the existence of User credentials whose names start with the `name` argument specified in the constructor.

The action string is converted to lowercase before processing.

Following is a Java 2 style policy entry which grants a user administration bundle a number of UserAdminPermissions:

```
grant codeBase "${jars}useradmin_console.jar" {
   permission org.osgi.service.useradmin.UserAdminPermission "admin";
```

All Page Within This Box

```
    permission org.osgi.service.useradmin.UserAdminPermission
               "com.foo.*", "changeProperty,getCredential,changeCredential";
    permission org.osgi.service.useradmin.UserAdminPermission
               "user.*", "changeProperty,changeCredential";
};
```

The first permission statement grants the bundle the permission to perform any UserAdmin operations of type "admin", that is, create and remove roles and configure Group roles.

The second permission statement grants the bundle the permission to change any properties as well as get and change any credentials whose names start with `com.foo.`.

The third permission statement grants the bundle the permission to change any properties and credentials whose names start with `user.`. This means that the bundle is allowed to change, but not retrieve any credentials with the given prefix.

The following policy entry empowers the Http bundle to perform user authentication:

```
grant codeBase "${jars}http.jar" {
  permission org.osgi.service.useradmin.UserAdminPermission
             "user.password", "getCredential";
};
```

The permission statement grants the Http bundle the permission to validate any password credentials (for authentication purposes), but the bundle is not allowed to change any properties or credentials.

See Also:

Serialized Form

Field Summary	
`static java.lang.String`	**`ADMIN`** The permission name "admin".
`static java.lang.String`	**`CHANGE_CREDENTIAL`** The action string "changeCredential".
`static java.lang.String`	**`CHANGE_PROPERTY`** The action string "changeProperty".
`static java.lang.String`	**`GET_CREDENTIAL`** The action string "getCredential".

Constructor Summary
`UserAdminPermission``(java.lang.String name, java.lang.String actions)` Creates a new UserAdminPermission with the specified name and actions.

Method Summary	
`boolean`	**`equals`**`(java.lang.Object obj)` Checks two UserAdminPermission objects for equality.
`java.lang.String`	**`getActions`**`()`

All Page Within This Box

	Returns the canonical string representation of the actions, separated by comma.
int	**hashCode**() Returns the hash code of this UserAdminPermission.
boolean	**implies**(java.security.Permission p) Checks if this UserAdminPermission object "implies" the specified permission.
java.security.PermissionCollection	**newPermissionCollection**() Returns a new PermissionCollection object for storing UserAdminPermission objects.
java.lang.String	**toString**() Returns a string describing this UserAdminPermission.

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

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

Field Detail

4.8.1 ADMIN

```
public static final java.lang.String ADMIN
```
The permission name "admin".

4.8.2 CHANGE_PROPERTY

```
public static final java.lang.String CHANGE_PROPERTY
```
The action string "changeProperty".

4.8.3 CHANGE_CREDENTIAL

```
public static final java.lang.String CHANGE_CREDENTIAL
```
The action string "changeCredential".

4.8.4 GET_CREDENTIAL

```
public static final java.lang.String GET_CREDENTIAL
```
The action string "getCredential".

Constructor Detail

All Page Within This Box

4.8.5 UserAdminPermission

`public` **`UserAdminPermission`**`(java.lang.String name,`
`java.lang.String actions)`

Creates a new UserAdminPermission with the specified name and actions. `name` is either the reserved string "admin" or the name of a credential or property, and `actions` contains a comma-separated list of the actions granted on the specified name. Valid actions are "changeProperty", "changeCredential", and "getCredential".

Parameters:
`name` - the name of this UserAdminPermission
`actions` - the action string.
Throws:
`java.lang.IllegalArgumentException` - If `name` equals "admin" and `actions` is not `null`.

Method Detail

4.8.6 equals

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

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

Overrides:
`equals` in class `java.security.BasicPermission`
Parameters:
`obj` - the object to be compared for equality with this object.
Returns:
`true` if `obj` is a UserAdminPermission, and has the same name and actions as this UserAdminPermission object.

4.8.7 getActions

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

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

Overrides:
`getActions` in class `java.security.BasicPermission`
Returns:
the canonical string representation of the actions.

4.8.8 hashCode

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

Returns the hash code of this UserAdminPermission.

Overrides:
`hashCode` in class `java.security.BasicPermission`

4.8.9 implies

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

Checks if this UserAdminPermission object "implies" the specified permission.

More specifically, this method returns true if:

All Page Within This Box

- *p* is an instanceof UserAdminPermission,
- *p*'s actions are a proper subset of this object's actions, and
- *p*'s name is implied by this object's name. For example, "java.*" implies "java.home".

Overrides:
`implies` in class `java.security.BasicPermission`
Parameters:
p - the permission to check against.
Returns:
`true` if the specified permission is implied by this object; `false` otherwise.

4.8.10 newPermissionCollection

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

Returns a new PermissionCollection object for storing UserAdminPermission objects.
Overrides:
`newPermissionCollection` in class `java.security.BasicPermission`
Returns:
a new PermissionCollection object suitable for storing UserAdminPermission objects.

4.8.11 toString

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

Returns a string describing this UserAdminPermission. The convention is to specify the class name, the permission name, and the actions in the following format: '("ClassName" "name" "actions")'.
Overrides:
`toString` in class `java.security.Permission`
Returns:
information about this Permission.

5 Security Considerations

The UserAdmin and related interfaces add authentication and authorization capabilities to OSGi gateways.

All Page Within This Box

6 Document Support

6.1 References

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

[2]. Howes, T., The String Representation of LDAP Search Filters, IETF RFC2254, December 1997.

6.2 Author's Address

Name	Ben Reed
Company	IBM
Address	
Voice	
e-mail	breed@almaden.ibm.com

Name	Jan Sparud
Company	Gatespace
Address	
Voice	
e-mail	sparud@gatespace.com

Name	Jan Luehe
Company	Sun Microsystems, Inc.
Address	901 San Antonio Road, M/S UCUP01-207 Palo Alto, CA 94303
Voice	+1 408 863 3216
e-mail	luehe@eng.sun.com

All Page Within This Box

6.3 Acronyms and Abbreviations

6.4 End of Document