RFC 139: JMX Control of OSGi

0.51 Draft

123 Pages

Abstract

OSGi defines a rich command and control API for the core system. This API, however, is not well suited for remote management systems, nor does it provide a useful definition for layering a JMX management interface around. This specification defines a JMX compliant model for managing the OSGi framework and critical compendium services. This model consists of a set of interfaces and simple types which provide a more suitable management and monitoring API for remote management applications.

Copyright © Oracle Corporation 2009

This contribution is made to the OSGi Alliance as MEMBER LICENSED MATERIALS pursuant to the terms of the OSGi Member Agreement and specifically the license rights and warranty disclaimers as set forth in Sections 3.2 and 12.1, respectively.

All company, brand and product names contained within this document may be trademarks that are the sole property of the respective owners.

The above notice must be included on all copies of this document that are made.

0 Document Information

0.1 Table of Contents

Copyright © Oracle Corporation 2009 All Rights Reserved

0.2 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 8.1.

```
Source code is shown in this typeface.
```

0.3 Revision History

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

Revision	Date	Comments
0.1 draft	*August 27th 2008*	First draft of this RFC *Hal Hildebrand, Oracle Corporation.* *hal.hildebrand@oracle.com*
0.2 draft	*September 2, 2008*	Incorporated comments from Richard Hall regarding naming, clarified use of symbolic name + version in arguments *Hal Hildebrand, Oracle Corporation.* *hal.hildebrand@oracle.com*
0.3 draft	*September 16, 2008*	Fully fleshed out java doc of API, changed much of the fine grained access to deal in the currency of opaque identifiers (i.e. JMX Object Name strings) rather than {bundleSymbolicName;version} pairs *Hal Hildebrand, Oracle Corporation.* *hal.hildebrand@oracle.com*
0.4 draft	*September 23, 2008*	Redesigned interfaces to eliminate individual Mbean representation for Bundles, Packages, and Services to allow the framework to effectively scale. Reformatted and transferred to Open Document *Hal Hildebrand, Oracle Corporation.* *hal.hildebrand@oracle.com*

Copyright © Oracle Corporation 2009 All Rights Reserved

Revision	Date	Comments
0.5 draft	*November 24, 2008*	Cleaned up interfaces and ensured all required parameters are in place. Added event types for BundleStateMBean and ServiceStateMBean. Added additional Runtime exceptions to indicate missing Bundles or Services *Hal Hildebrand, Oracle Corporation. hal.hildebrand@oracle.com*
0.51 draft	*November 25, 2008*	Added more documentation on the structure of the CompositeData types returned by the various operations of the JMX MBeans. Created a new package of classes which provide the Coding/Decoding of these CompositeData types and serve as the documentation of their structure. *Hal Hildebrand, Oracle Corporation. hal.hildebrand@oracle.com*
0.52 draft	*March 3, 2009*	Updated the Java Doc to reflect the final state of the RI *Hal Hildebrand, Oracle Corporation. hal.hildebrand@oracle.com*

1 Introduction

The Java Management Extensions is the standard API specification for providing a management interface to J2SE and JEE applications. The JMX specification defines the design patterns, APIs, services and architecture for application and network management and monitoring in the Java programming language. The need to administer, monitor and manage a container is now recognized as a prerequisite in the enterprise software domain.

While OSGi defines a rich API for controlling all aspects of the framework, this API is not suitable for direct usage in the JMX framework. This document describes a proposal for an interface adaption of the existing OSGi framework which can be used to expose the OSGi framework manipulation API to any JMX compliant implementation. In addition, interfaces and system semantics for a monitoring system are proposed for exposing the underlying artifacts of the OSGi framework such as Services, Bundles, Packages, etc. Finally, a standardized JMX object naming standard is proposed so that management objects are uniformly named across implementations such that any JMX compliant system can find, manipulate and interact with the framework and artifacts that it manages.

It is important to note that JMX is not a remote communication standard. Rather, JMX is a specification which defines how arbitrary remote communication protocols and mechanisms can be adapted to interact with and control the underlying management apis exposed by compliant implementations.

The JMX architecture is composed of three levels:

- Instrumentation
- Agent
- Remote Management

At the instrumentation layer, the resources of the system are instrumented using Managed Beans which expose their management interfaces through a JMX agent for remote management an monitoring. The JMX agent layer is mainly represented by the MBean server. This is the managed object server where the MBeans are registered.

Copyright © Oracle Corporation 2009 All Rights Reserved

The JMX agent includes a set of service for manipulating the MBeans, directly controls the resources and makes them available to remote management agents. The remote management layer provides the specification for the actual remote communication protocol adapters and defines standard connectors which make the JMX agent accessible to remote management applications outside of the Java process.

2 Application Domain

The primary domain addressed by this RFC is the management space of enterprise Java applications, although a solution to the requirements raised by the RFC should prove useful in other management domains.

3 Problem Description

Enterprise middleware includes infrastructure for manageability based on the Java Management Extensions framework. JMX is so widespread in the Java enterprise space that it has now become mandatory to expose management functions of middleware through standardized JMX APIs for the component. Consequently, if OSGi wishes to participate in modern enterprise management systems, exposing a JMX compliant management interface for the framework becomes a mandatory step to participating in the enterprise middleware space

3.1 Integration with Existing Enterprise Management Systems

A significant number of Java enterprise middleware systems require JMX compliance in order to integrate into the management framework for the middleware platform. OSGi has an advantage over other containers in that the control over the base framework is both robust and well defined. Exposing this API through JMX allows these management systems to obtain high fidelity control in a framework independent fashion. This allows the OSGi container to integrate with the existing enterprise platform and participate fully in the solution space.

3.2 Integration with JMX Management Systems

JMX is the management standard for the Java standard edition, as well as the standard for managing the enterprise middleware space. JMX has turned out to be a popular standard in an endless number of application domains and providing a JMX compliant management and monitoring API would allow OSGi to seamlessly integrate with these management strategies as well.

3.3 High Fidelity Remote Control

It is quite common to require the control of an OSGi framework from outside the framework itself. The controlling system need not be “remote” in the traditional sense of the word, but could simply be in another class loading space within the same Java process. A carefully designed JMX API for the OSGi framework would allow systems to manipulate and control the OSGi container.

Copyright © Oracle Corporation 2009 All Rights Reserved

3.4 Security: Authentication and Access Control

Exposing any system remotely opens up a new, potentially, devastating security hole in a system. Remote entities should establish their identity and the management system should be able to control the access these entities have over the management system. JMX seamlessly interoperates with the Java Authentication and Authorization Service (JAAS) and Java 2 platform Standard Edition (J2SE) Security Architecture.

3.5 Localization Issues

Dealing with the intricacies and perils of localization is something that is expected by the customers of modern enterprise middleware platforms. JMX integrates well with existing Java mechanisms for dealing with the thorny issues involved in localization as well as defining it's own sophisticated mechanisms for the localization issues created by having remote users to the framework, each of which may have different localization requirements.

4 Requirements

- The solution MUST be compliant to the Java Management Extensions specification.
- The solution MUST exploit Java 2 security if enabled by complying with the standard JMX security and authentication mechanisms.
- The solution MUST be remotely manageable through JMX compliant connectors and adapters.
- The solution SHOULD provide JMX models for all the core framework API, artifacts and services.
- The solution MUST define a uniform JMX object naming convention and format for uniquely identifying all JMX managed artifacts.
- The solution MUST provide access to all core OSGi framework APIs and services using primitive types and simple JRE classes such as Dictionary to allow the system to be used without requiring additional classes on the JMX client.
- The solution MAY define a handful of additional types for optional APIs which deal with complicated, aggregated framework state such as Bundles, Services and Packages.
- The solution SHOULD define a JMX integration of critical compendium services such as the configuration administration, logging, event administration and preferences services.
- The solution SHOULD allow the dynamic update of the JMX environment's view of the OSGi framework state through the use of OSGi and JMX provided callbacks and listeners.
- The solution SHOULD be as simple as is possible, making use of no JMX specific artifacts to accomplish the goal, so that non JMX management frameworks may reuse of the resulting framework without pulling in JMX.
- The solution SHOULD NOT define any new models for manipulating the OSGi framework other than those already present in the existing framework APIs.
- The solution SHOULD NOT define a generic framework for JMX integration of arbitrary OSGi services. The goal is to manage the OSGi framework through JMX, not provide a generic mechanism that be used to expose management of arbitrary OSGi services through JMX.

Copyright © Oracle Corporation 2009 All Rights Reserved

5 Technical Solution

5.1 Architectural Overview

A set of Java interfaces defining what are known in JMX as Standard MBeans are created which closely follow the underlying OSGi command and control API. These interfaces define the manageable attributes and operations exposed through the JMX API. Standard MBeans were chosen following JMX best practices. Specifying these APIs using Standard MBeans allows them to be documented using the familiar Javadoc tool, and it allows client code to interact with them straightforwardly via proxies, using MBeanServerInvocationHandler. Contrast the code without a proxy:

```
MBeanServer mbs = ...;
Integer sizeI = (Integer) mbs.getAttribute(objectName, "Size");
int size = sizeI.intValue();
if (size > desiredSize) {
    mbs.invoke(objectName,
               dropOldest",
               new Integer[] {new Integer(size - desiredSize)},
                             new String[] {"int"});}
}
```

with the code that uses a proxy:

```
MBeanServer mbs = ...;
CacheControlMBean cacheControl = (CacheControlMBean)
 MbeanServerInvocationHandler.newProxyInstance(mbs,
                                               objectName,
                                               CacheControlMBean.class,
                                               false);
int size = cacheControl.getSize();
if (size > desiredSize)
     cacheControl.dropOldest(size - desiredSize);
```

The creation of the proxy is somewhat verbose, but once it is available, the MBean can be accessed like a local object. This is much easier to write and read, and much less error-prone, than accessing the MBeanServer method directly.

All managed objects in JMX are referenced via the JMX *Object Names*. JMX Object Names are strings which can be resolved within the context of a JMX MBeanServer in order to invoke operations on the MBean referred to by the name. In this RFC, all interfaces are specified to return opaque Strings rather than actual JMX Object Names so that the MBean interfaces contain no JMX specific artifacts and can be used with a variety of remote access protocols such as SNMP, etc. Non JMX use of these APIs can use these Strings as their own opaque identifiers without any change to the interfaces themselves.

5.2 The MBeanServer

The construction, maintenance and lifecycle of the MBeanServer which will host the MBeans defined in this specification is intentionally left undefined. It is left undefined as the MBeanServer is invariably tied to the particular application that is responsible for it. For example, the MBeanServer may exist outside the OSGi framework that the MBeans are managing. Or there may be multiple MBean servers which contain the MBeans defined in this specification. The introduction of nested frameworks, such as those defined in RFC 138, may have their management MBeans hosted in the MBeanServer which hosts the MBeans for the outermost OSGi container. Alternatively, these MBeans may be hosted, for example, in an application server's MBeanServer which is embedding mulitple OSGi containers.

Copyright © Oracle Corporation 2009 All Rights Reserved

5.3 Interfaces

The API is divided into 2 interface sets. The first defines the management interface for the OSGi core command and control API. The second defines the management interface for essential compendium service APIs.

Note on JMX interface naming conventions wrt Standard MBeans: When using JMX Standard MBeans, the standard states that you have a Class which represents the actual implementation and the interface which is used by the JMX infrastructure. Standard MBeans work off of a naming convention where the interface the implementation implements is named <BaseClass>.<MBean>. Thus, the JMX class which implements the BundleStateMBean is named BundleState.

5.4 JMX Object Names

This RFC defines 8 JMX MBeans. These MBeans are bound to JMX ObjectNames within a MBeanServer. The standard ObjectNames for these MBeans are:

MBean	JMX ObjectName
FrameworkMBean	osgi.core::type=framework
BundleStateMBean	osgi.core:type=bundleState
PackageStateMBean	osgi.core:type=packageState
ServiceStateMBean	osgi.core:type=serviceState
ConfigAdminManagerMBean	osgi.compendium:service=configAdminManager
PermissionManagerMBean	osgi.compendium:service=permissionAdmin
ProvisioningMBean	osgi.compendium:service=provisioningService
UserManagerMBean	osgi.compendium:service=userAdmin

5.5 CompositeData types

Some of the operations in this API return JMX open data types represented by CompositeData. These composite data types are facilitated by a set of codec classes which serve to document the structure of the Composite data as well as provide a convenient mechanism to represent the CompositeData type within Java as well as the conversion to and from the composite data type.

5.5.1 BundleBatchActionResult

org.osgi.jmx.codec **Class BundleBatchActionResult**

```
Object
  org.osgi.jmx.codec.BundleBatchActionResult

public class BundleBatchActionResult
extends Object
```

This class represents the CODEC for the resulting composite data from the batch operations on the bundles in the FrameworkMBean. It serves as both the documentation of the type structure and as the codification of the mechanism to convert to/from the CompositeData.

Copyright © Oracle Corporation 2009 All Rights Reserved

The structure of the composite data is:

Success	Boolean
Error	String
Completed	Array of long
BundleInError	long
Remaining	Array of long

Field Summary

static CompositeType	**RESULT** The CompositeType which represents the result of batch operations on the FrameworkMBean

Constructor Summary

BundleBatchActionResult() Construct a result signifying the successful completion of the batch operation.
BundleBatchActionResult(CompositeData compositeData) Construct a result representing the contents of the supplied CompositeData returned from a batch operation.
BundleBatchActionResult(String errorMessage, long[] completed, long bundleInError, long[] remaining) Construct a result indictating the failure of a batch operation.

Method Summary

CompositeData	**asCompositeData**() Answer the receiver encoded as CompositeData
long	**getBundleInError**() Answer the bundle identifier which indicates the bundle that produced an error during the batch operation.
long[]	**getCompleted**() If the operation failed, answer the list of bundle identifiers that successfully completed the batch operation.
String	**getErrorMessage**() Answer the error message indicating the error that occurred during the batch operation or null, if the operation was a success.
long[]	**getRemaining**() If the operation was unsuccessful, answer the list of bundle identifiers of the bundles that were not processed during the batch operation.

Copyright © Oracle Corporation 2009 All Rights Reserved

boolean	**isSuccess**() Answer true if the batch operation was successful, false otherwise.

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

Field Detail

RESULT

```
public static final CompositeType RESULT
```
The CompositeType which represents the result of batch operations on the FrameworkMBean

Constructor Detail

BundleBatchActionResult

```
public BundleBatchActionResult()
```
Construct a result signifying the successful completion of the batch operation.

BundleBatchActionResult

```
public BundleBatchActionResult(CompositeData compositeData)
```
Construct a result representing the contents of the supplied CompositeData returned from a batch operation.

Parameters:
`compositeData` - - the CompositeData representing the result of a batch operation.

BundleBatchActionResult

```
public BundleBatchActionResult(String errorMessage,
                               long[] completed,
                               long bundleInError,
                               long[] remaining)
```
Construct a result indictating the failure of a batch operation.

Parameters:
`errorMessage` - - the message indicating the error

Copyright © Oracle Corporation 2009 All Rights Reserved

`completed` - - the list of bundle identifiers indicating bundles that have successfully completed the batch operation
`bundleInError` - - the identifier of the bundle which produced the error
`remaining` - - the list of bundle identifiers which remain unprocessed

Method Detail

asCompositeData

`public CompositeData asCompositeData()`
Answer the receiver encoded as CompositeData

Returns:
the CompositeData encoding of the receiver.

getBundleInError

`public long getBundleInError()`
Answer the bundle identifier which indicates the bundle that produced an error during the batch operation.

Returns:
the bundle identifier of the bundle in error, or -1L if no error occurred

getCompleted

`public long[] getCompleted()`
If the operation failed, answer the list of bundle identifiers that successfully completed the batch operation. If the operation was successful, then the list is null;

Returns:
the list of bundle identifiers or null if the operation was successful

getErrorMessage

`public String getErrorMessage()`
Answer the error message indicating the error that occurred during the batch operation or null, if the operation was a success.

Returns:
the String error message

getRemaining

`public long[] getRemaining()`

Copyright © Oracle Corporation 2009 All Rights Reserved

If the operation was unsuccessful, answer the list of bundle identifiers of the bundles that were not processed during the batch operation. If the operation was a success, then answer null

Returns:
the remaining bundle identifiers or null if the operation was a success

isSuccess

`public boolean isSuccess()`
Answer true if the batch operation was successful, false otherwise.

Returns:
the success of the batch operation

5.5.2 BundleBatchInstallResult

org.osgi.jmx.codec Class BundleBatchInstallResult

```
Object
  org.osgi.jmx.codec.BundleBatchInstallResult
```

```
public class BundleBatchInstallResult
extends Object
```

This class represents the CODEC for the resulting composite data from the batch install operations on the bundles in the FrameworkMBean. It serves as both the documentation of the type structure and as the codification of the mechanism to convert to/from the CompositeData.

The structure of the composite data is:

Success	Boolean
Error	String
Completed	Array of long
BundleInError	String
Remaining	Array of String

Field Summary	
static CompositeType	**BATCH_RESULT** The CompositeType which represents the result of batch install operations on the FrameworkMBean

Copyright © Oracle Corporation 2009 All Rights Reserved

Constructor Summary

BundleBatchInstallResult(CompositeData compositeData) Construct a result representing the contents of the supplied CompositeData returned from a batch operation.
BundleBatchInstallResult(long[] completed) Construct a result signifying the successful completion of the batch operation.
BundleBatchInstallResult(String errorMessage, long[] completed, String bundleInError, String[] remaining) Construct a result indictating the failure of a batch operation.

Method Summary

CompositeData	**asCompositeData**() Answer the receiver encoded as CompositeData
String	**getBundleInError**() Answer the bundle location which indicates the bundle that produced an error during the batch operation.
long[]	**getCompleted**() Answer the list of bundle identifiers that successfully completed the batch operation.
String	**getErrorMessage**() Answer the error message indicating the error that occurred during the batch operation or null if the operation was successful
String[]	**getRemaining**() Answer the list of locations of the bundles that were not processed during the batch operation, or null if the operation was successsful
boolean	**isSuccess**() Answer true if the batch operation was successful, false otherwise.

Methods inherited from class Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

BATCH_RESULT

```
public static final CompositeType BATCH_RESULT
```

The CompositeType which represents the result of batch install operations on the FrameworkMBean

Copyright © Oracle Corporation 2009 All Rights Reserved

Constructor Detail

BundleBatchInstallResult

```
public BundleBatchInstallResult(CompositeData compositeData)
```
Construct a result representing the contents of the supplied CompositeData returned from a batch operation.

Parameters:
`compositeData` - - the CompositeData representing the result of a batch operation.

BundleBatchInstallResult

```
public BundleBatchInstallResult(long[] completed)
```
Construct a result signifying the successful completion of the batch operation.

Parameters:
`completed` - - the resulting bundle identifiers of the installed bundles

BundleBatchInstallResult

```
public BundleBatchInstallResult(String errorMessage,
                                long[] completed,
                                String bundleInError,
                                String[] remaining)
```
Construct a result indictating the failure of a batch operation.

Parameters:
`errorMessage` - - the message indicating the error
`completed` - - the list of bundle identifiers indicating bundles that have successfully completed the batch operation
`bundleInError` - - the identifier of the bundle which produced the error
`remaining` - - the list of bundle identifiers which remain unprocessed

Method Detail

asCompositeData

```
public CompositeData asCompositeData()
```
Answer the receiver encoded as CompositeData

Returns:
the CompositeData encoding of the receiver.

getBundleInError

Copyright © Oracle Corporation 2009 All Rights Reserved

`public String` **`getBundleInError()`**
Answer the bundle location which indicates the bundle that produced an error during the batch operation.

Returns:
the bundle location of the bundle in error, or null if no error occurred

getCompleted

`public long[]` **`getCompleted()`**
Answer the list of bundle identifiers that successfully completed the batch operation. If the operation was unsuccessful, this will be a partial list. If this operation was successful, this will be the full list of bundle ids. This list corresponds one to one with the supplied list of bundle locations provided to the batch install operations.

Returns:
the list of identifiers of the bundles that successfully installed

getErrorMessage

`public String` **`getErrorMessage()`**
Answer the error message indicating the error that occurred during the batch operation or null if the operation was successful

Returns:
the String error message if the operation was unsuccessful, or null if the operation was successful

getRemaining

`public String[]` **`getRemaining()`**
Answer the list of locations of the bundles that were not processed during the batch operation, or null if the operation was successsful

Returns:
the remaining bundle locations if the operation was successful, or null if the operation was unsuccsesful.

isSuccess

`public boolean` **`isSuccess()`**
Answer true if the batch operation was successful, false otherwise.

Returns:
the success of the batch operation

Copyright © Oracle Corporation 2009 All Rights Reserved

5.5.3 OSGiAuthorization

org.osgi.jmx.codec Class OSGiAuthorization

```
Object
  org.osgi.jmx.codec.OSGiAuthorization
```

```
public class OSGiAuthorization
extends Object
```

Field Summary	
static CompositeType	**AUTHORIZATION**
protected String	**name**
protected String[]	**roles**

Constructor Summary
OSGiAuthorization(Authorization authorization)
OSGiAuthorization(CompositeData data)
OSGiAuthorization(String name, String[] roles)

Method Summary	
CompositeData	**asCompositeData**()
String	**getName**()
String[]	**getRoles**()

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

Field Detail

name

```
protected String name
```

roles

Copyright © Oracle Corporation 2009 All Rights Reserved

```
protected String[] roles
```

AUTHORIZATION

```
public static final CompositeType AUTHORIZATION
```

Constructor Detail

OSGiAuthorization

```
public OSGiAuthorization(CompositeData data)
```

OSGiAuthorization

```
public OSGiAuthorization(Authorization authorization)
```

OSGiAuthorization

```
public OSGiAuthorization(String name,
                         String[] roles)
```

Method Detail

asCompositeData

```
public CompositeData asCompositeData()
                                                                  throws
OpenDataException
```

Throws:
OpenDataException

getName

```
public String getName()
```

Returns:
the name

getRoles

```
public String[] getRoles()
```

Returns:
the roles

Copyright © Oracle Corporation 2009 All Rights Reserved

5.5.4 OSGiBundle

org.osgi.jmx.codec Class OSGiBundle

```
Object
  org.osgi.jmx.codec.OSGiBundle
```

```
public class OSGiBundle
extends Object
```

This class represents the CODEC for the composite data representing a single OSGi Bundle.

It serves as both the documentation of the type structure and as the codification of the mechanism to convert to/from the CompositeData.

The structure of the composite data is:

Location	String
Identifier	long
SymbolicName	String
StartLevel	int
State	String
LastModified	long
PersistentlyStarted	boolean
RemovalPending	boolean
Required	boolean
Fragment	boolean
RegisteredServices	Array of long
ServicesInUse	Array of long
Headers	TabularData of Key/Value String pairs
ExportedPackages	Array of String
ImportedPackages	Array of String
Fragments	Array of long
Hosts	Array of long
RequiredBundles	Array of long
RequiringBundles	Array of long

Field Summary

static CompositeType	**BUNDLE** The CompositeType which represents a single OSGi bundle
static CompositeType	**BUNDLE_HEADER** The CompositeType which represents a key/value header pair
static TabularType	**BUNDLE_HEADER_TABLE**

Copyright © Oracle Corporation 2009 All Rights Reserved

	The TabularType which represents the map of bundle headers
static TabularType	**BUNDLE_TABLE** The TabularType which represents a list of bundles

Constructor Summary

OSGiBundle(BundleContext bc, PackageAdmin admin, StartLevel sl, Bundle b) Construct an OSGiBundle representation
OSGiBundle(CompositeData data) Construct an OSGiBundle from the encoded CompositeData
OSGiBundle(String location, long identifier, String symbolicName, int startLevel, String state, long lastModified, boolean persistentlyStarted, boolean removalPending, boolean required, boolean fragment, long[] registeredServices, long[] servicesInUse, java.util.Map<String,String> headers, String[] exportedPackages, String[] importedPackages, long[] fragments, long[] hosts, long[] requiredBundles, long[] requiringBundles) Construct and OSGiBundle

Method Summary

CompositeData	**asCompositeData**() Answer the receiver encoded as CompositeData
String[]	**getExportedPackages**()
long[]	**getFragments**()
java.util.Map<String,String>	**getHeaders**()
long[]	**getHosts**()
long	**getIdentifier**()
String[]	**getImportedPackages**()
long	**getLastModified**()
String	**getLocation**()
long[]	**getRegisteredServices**()
long[]	**getRequiredBundles**()
long[]	**getRequiringBundles**()

Copyright © Oracle Corporation 2009 All Rights Reserved

long[]	**getServicesInUse**()
int	**getStartLevel**()
String	**getState**()
String	**getSymbolicName**()
static TabularData	**headerTable**(Bundle b) Answer the TabularData representing the list of bundle headers for a bundle
static TabularData	**headerTable**(java.util.Map<String,String> headers)
boolean	**isFragment**()
boolean	**isPersistentlyStarted**()
boolean	**isRemovalPending**()
boolean	**isRequired**()
static TabularData	**tableFrom**(java.util.ArrayList<OSGiBundle> bundles) Answer the TabularData representing the list of OSGiBundle state

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

Field Detail

BUNDLE_HEADER

public static final CompositeType **BUNDLE_HEADER**
The CompositeType which represents a key/value header pair

BUNDLE_HEADER_TABLE

public static final TabularType **BUNDLE_HEADER_TABLE**
The TabularType which represents the map of bundle headers

Copyright © Oracle Corporation 2009 All Rights Reserved

BUNDLE

```
public static final CompositeType BUNDLE
```
The CompositeType which represents a single OSGi bundle

BUNDLE_TABLE

```
public static final TabularType BUNDLE_TABLE
```
The TabularType which represents a list of bundles

Constructor Detail

OSGiBundle

```
public OSGiBundle(CompositeData data)
```
Construct an OSGiBundle from the encoded CompositeData

Parameters:
`data` - - the encoded representation of the bundle

OSGiBundle

```
public OSGiBundle(BundleContext bc,
                  PackageAdmin admin,
                  StartLevel sl,
                  Bundle b)
```
Construct an OSGiBundle representation

Parameters:
`bc` - - the BundleContext to be used.
`admin` - - the PackageAdmin service
`sl` - - the StartLevel service
`b` - - the Bundle to represent

OSGiBundle

```
public OSGiBundle(String location,
                  long identifier,
                  String symbolicName,
                  int startLevel,
```

Copyright © Oracle Corporation 2009 All Rights Reserved

```
                String state,
                long lastModified,
                boolean persistentlyStarted,
                boolean removalPending,
                boolean required,
                boolean fragment,
                long[] registeredServices,
                long[] servicesInUse,
                java.util.Map<String,String> headers,
                String[] exportedPackages,
                String[] importedPackages,
                long[] fragments,
                long[] hosts,
                long[] requiredBundles,
                long[] requiringBundles)
```

Construct and OSGiBundle

Parameters:
`location` -
`identifier` -
`symbolicName` -
`startLevel` -
`state` -
`lastModified` -
`persistentlyStarted` -
`removalPending` -
`required` -
`fragment` -
`registeredServices` -
`servicesInUse` -
`headers` -
`exportedPackages` -
`importedPackages` -
`fragments` -
`hosts` -
`requiredBundles` -
`requiringBundles` -

Method Detail

tableFrom

```
public static TabularData
tableFrom(java.util.ArrayList<OSGiBundle> bundles)
```

Answer the TabularData representing the list of OSGiBundle state

Parameters:
`bundles` - - the list of bundles to represent
Returns:
the Tabular data which represents the list of bundles

Copyright © Oracle Corporation 2009 All Rights Reserved

headerTable

```
public static TabularData headerTable(Bundle b)
```
Answer the TabularData representing the list of bundle headers for a bundle

Parameters:
b -
Returns:
the bundle headers

headerTable

```
public static TabularData
headerTable(java.util.Map<String,String> headers)
```

asCompositeData

```
public CompositeData asCompositeData()
```
Answer the receiver encoded as CompositeData

Returns:
the CompositeData encoding of the receiver.

getExportedPackages

```
public String[] getExportedPackages()
```
Returns:
The list of exported packages by this bundle, in the form of ;

getFragments

```
public long[] getFragments()
```
Returns:
the list of identifiers of the bundle fragments which use this bundle as a host

getHeaders

```
public java.util.Map<String,String> getHeaders()
```
Returns:
the map of headers for this bundle

getHosts

```
public long[] getHosts()
```

Copyright © Oracle Corporation 2009 All Rights Reserved

Returns:
list of identifiers of the bundles which host this fragment

getIdentifier

`public long` **`getIdentifier()`**
Returns:
the identifier of this bundle

getImportedPackages

`public String[]` **`getImportedPackages()`**
Returns:
The list of imported packages by this bundle, in the form of ;

getLastModified

`public long` **`getLastModified()`**
Returns:
the last modified time of this bundle

getLocation

`public String` **`getLocation()`**
Returns:
the name of this bundle

getRegisteredServices

`public long[]` **`getRegisteredServices()`**
Returns:
the list of identifiers of the services registered by this bundle

getRequiredBundles

`public long[]` **`getRequiredBundles()`**
Returns:
the list of identifiers of bundles required by this bundle

getRequiringBundles

`public long[]` **`getRequiringBundles()`**
Returns:
the list of identifiers of bundles which require this bundle

getServicesInUse

Copyright © Oracle Corporation 2009 All Rights Reserved

`public long[] getServicesInUse()`
Returns:
the list of identifiers of services in use by this bundle

getStartLevel

`public int getStartLevel()`
Returns:
the start level of this bundle

getState

`public String getState()`
Returns:
the state of this bundle

getSymbolicName

`public String getSymbolicName()`
Returns:
the symbolic name of this bundle

isFragment

`public boolean isFragment()`
Returns:
true if this bundle represents a fragment

isPersistentlyStarted

`public boolean isPersistentlyStarted()`
Returns:
true if this bundle is persistently started

isRemovalPending

`public boolean isRemovalPending()`
Returns:
true if this bundle is pending removal

isRequired

`public boolean isRequired()`
Returns:
true if this bundle is required

Copyright © Oracle Corporation 2009 All Rights Reserved

5.5.5 OSGiBundleEvent

org.osgi.jmx.codec Class OSGiBundleEvent

```
Object
  org.osgi.jmx.codec.OSGiBundleEvent

public class OSGiBundleEvent
extends Object
```

Author:
Hal Hildebrand Date: Nov 24, 2008 Time: 2:22:34 PM
This class represents the CODEC for the composite data representing a OSGi BundleEvent

It serves as both the documentation of the type structure and as the codification of the mechanism to convert to/from the CompositeData.

The structure of the composite data is:

Identifier	long
location	String
SymbolicName	String
EventType	int

Field Summary

static CompositeType	**BUNDLE_EVENT** The CompositeType representation of the event

Constructor Summary

OSGiBundleEvent(BundleEvent event) Construct an OSGiBundleEvent from the supplied BundleEvent
OSGiBundleEvent(CompositeData data) Construct an OSGiBundleEvent from the CompositeData representing the event
OSGiBundleEvent(long bundleId, String location, String symbolicName, int eventType) Construct the OSGiBundleEvent

Method Summary

CompositeData	**asCompositeData**() Answer the receiver encoded as CompositeData
long	**getBundleId**()
int	**getEventType**()

Copyright © Oracle Corporation 2009 All Rights Reserved

String	getLocation()
String	getSymbolicName()

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

Field Detail

BUNDLE_EVENT

```
public static final CompositeType BUNDLE_EVENT
```
The CompositeType representation of the event

Constructor Detail

OSGiBundleEvent

```
public OSGiBundleEvent(BundleEvent event)
```
Construct an OSGiBundleEvent from the supplied BundleEvent

Parameters:
`event` - - the event to represent

OSGiBundleEvent

```
public OSGiBundleEvent(CompositeData data)
```
Construct an OSGiBundleEvent from the CompositeData representing the event

Parameters:
`data` - - the CompositeData representing the event.

OSGiBundleEvent

```
public OSGiBundleEvent(long bundleId,
                       String location,
                       String symbolicName,
                       int eventType)
```
Construct the OSGiBundleEvent

Copyright © Oracle Corporation 2009 All Rights Reserved

Parameters:
`bundleId` -
`location` -
`symbolicName` -
`eventType` -

Method Detail

asCompositeData

```
public CompositeData asCompositeData()
```
Answer the receiver encoded as CompositeData

Returns:
the CompositeData encoding of the receiver.

getBundleId

```
public long getBundleId()
```
Returns:
the identifier of the bundle for this event

getEventType

```
public int getEventType()
```
Returns:
the type of the event

getLocation

```
public String getLocation()
```
Returns:
the location of the bundle for this event

getSymbolicName

```
public String getSymbolicName()
```
Returns:
the symbolic name of the bundle for this event

5.5.6 OSGiGroup

org.osgi.jmx.codec **Class OSGiGroup**

```
Object
  org.osgi.jmx.codec.OSGiGroup
```

```
public class OSGiGroup
```

Copyright © Oracle Corporation 2009 All Rights Reserved

extends Object

Field Summary

CompositeType	**GROUP**
protected String[]	**members**
protected String[]	**requiredMembers**
protected OSGiUser	**user**

Constructor Summary

OSGiGroup(CompositeData data)
OSGiGroup(Group group)

Method Summary

CompositeData	**asCompositeData**()
String[]	**getMembers**()
String[]	**getRequiredMembers**()
OSGiUser	**getUser**()

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

Field Detail

user

protected OSGiUser **user**

members

protected String[] **members**

Copyright © Oracle Corporation 2009 All Rights Reserved

requiredMembers

```
protected String[] requiredMembers
```

GROUP

```
public final CompositeType GROUP
```

Constructor Detail

OSGiGroup

```
public OSGiGroup(Group group)
```

OSGiGroup

```
public OSGiGroup(CompositeData data)
```

Method Detail

asCompositeData

```
public CompositeData asCompositeData()
                                     throws
OpenDataException
```

Throws:
OpenDataException

getUser

```
public OSGiUser getUser()
```

Returns:
the user

getMembers

```
public String[] getMembers()
```

Returns:
the members

getRequiredMembers

```
public String[] getRequiredMembers()
```

Returns:
the requiredMembers

Copyright © Oracle Corporation 2009 All Rights Reserved

5.5.7 OSGiPackage

org.osgi.jmx.codec Class OSGiPackage

```
Object
  org.osgi.jmx.codec.OSGiPackage
```

```
public class OSGiPackage
extends Object
```

This class represents the CODEC for the composite data representing an OSGi ExportedPackage

It serves as both the documentation of the type structure and as the codification of the mechanism to convert to/from the CompositeData.

The structure of the composite data is:

Name	String
Version	String
PendingRemoval	boolean
BundleIdentifier	long
ImportingBundles	Array of long

Field Summary

static CompositeType	**PACKAGE** The CompositeType representation of the package
static TabularType	**PACKAGE_TABLE** The TabularType representation of a list of packages

Constructor Summary

OSGiPackage(CompositeData data) Construct an OSGiPackage from the encoded CompositeData
OSGiPackage(ExportedPackage pkg) Construct an OSGiPackage from the ExporetedPackage
OSGiPackage(String name, String version, boolean removalPending, long exportingBundle, long[] importingBundles) Construct and OSGiPackage from the supplied data

Method Summary

CompositeData	**asCompositeData**() Answer the receiver encoded as CompositeData

Copyright © Oracle Corporation 2009 All Rights Reserved

long	**getExportingBundle**()
long[]	**getImportingBundles**()
String	**getName**()
String	**getVersion**()
boolean	**isRemovalPending**()
static TabularData	**tableFrom**(java.util.ArrayList<OSGiPackage> packages)

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

Field Detail

PACKAGE

public static final CompositeType **PACKAGE**
The CompositeType representation of the package

PACKAGE_TABLE

public static final TabularType **PACKAGE_TABLE**
The TabularType representation of a list of packages

Constructor Detail

OSGiPackage

public **OSGiPackage**(CompositeData data)
Construct an OSGiPackage from the encoded CompositeData

Parameters:
data - - the CompositeData encoding the OSGiPackage

OSGiPackage

Copyright © Oracle Corporation 2009 All Rights Reserved

```
public OSGiPackage(ExportedPackage pkg)
```
Construct an OSGiPackage from the ExporetedPackage

Parameters:
pkg - - the ExporetedPackage

OSGiPackage

```
public OSGiPackage(String name,
                   String version,
                   boolean removalPending,
                   long exportingBundle,
                   long[] importingBundles)
```
Construct and OSGiPackage from the supplied data

Parameters:
name -
version -
removalPending -
exportingBundle -
importingBundles -

Method Detail

tableFrom

```
public static TabularData
tableFrom(java.util.ArrayList<OSGiPackage> packages)
```

asCompositeData

```
public CompositeData asCompositeData()
```
Answer the receiver encoded as CompositeData

Returns:
the CompositeData encoding of the receiver.

getExportingBundle

```
public long getExportingBundle()
```
Returns:
the identifier of the exporting bundle

getImportingBundles

```
public long[] getImportingBundles()
```

Copyright © Oracle Corporation 2009 All Rights Reserved

Returns:
the list of identifiers of the bundles importing this package

getName

```
public String getName()
```
Returns:
the name of the package

getVersion

```
public String getVersion()
```
Returns:
the version of the package

isRemovalPending

```
public boolean isRemovalPending()
```
Returns:
true if the package is pending removal

5.5.8 OSGiProperties

org.osgi.jmx.codec Class OSGiProperties

```
Object
  org.osgi.jmx.codec.OSGiProperties
```

```
public class OSGiProperties
extends Object
```

This class serves as both the documentation of the type structure and as the codification of the mechanism to convert to/from the TabularData.

This class represents the CODEC for property dictionaries. As JMX is a rather primitive system and is not intended to be a generic RMI type system, the set of types that can be transfered between the management agent and the managed OSGi container is limited to simple types, arrays of simple types and vectors of simple types. This enforcement is strict and no attempt is made to create a yet another generic serialization mechanism for transferring property values outside of these types.

The syntax for the type indicator

```
type   ::=    scalar | vector | array
scalar ::=    String | Integer | Long | Float |
              Double | Byte | Short | Character |
              Boolean | BigDecimal | BigInteger
primitive ::= int | long | float | double | byte | short |
              char | boolean
```

Copyright © Oracle Corporation 2009 All Rights Reserved

```
array ::=     <Array of primitive> | <Array of scalar>
vector ::=    Vector of scalar
```

The values for Arrays and Vectors are separated by ",".
The structure of the composite data for a row in the table is:

Key	String
Value	String
Type	String

The

Field Summary

static String	**KEY**
protected static java.util.Set<String>	**PRIMITIVE_TYPES**
static String[]	**PROPERTIES**
static CompositeType	**PROPERTY**
static TabularType	**PROPERTY_TABLE**
protected static java.util.Set<String>	**SCALAR_TYPES**
static String	**TYPE**
static String	**VALUE**

Constructor Summary

OSGiProperties()

Method Summary

protected static Object[]	**createScalarArray**(String type, int size)
static CompositeData	**encode**(String key, Object value)
protected static CompositeData	**encodeArray**(String key, Object value, Class<?> componentClazz)
protected static CompositeData	**encodeVector**(String key,

Copyright © Oracle Corporation 2009 All Rights Reserved

	java.util.Vector value)
static Object	**parse**(String value, String type)
protected static Object	**parseArray**(String value, java.util.StringTokenizer tokens)
protected static Object	**parsePrimitiveArray**(String value, String type)
protected static Object	**parseScalar**(String value, String type)
protected static Object	**parseScalarArray**(String value, String type)
protected static Object	**parseVector**(String value, java.util.StringTokenizer tokens)
static java.util.Hashtable<String,Object>	**propertiesFrom**(TabularData table)
protected static CompositeData	**propertyData**(String key, String value, String type)
static TabularData	**tableFrom**(java.util.Dictionary properties)
static TabularData	**tableFrom**(ServiceReference ref)
protected static String	**typeOf**(Class<?> clazz)

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

Field Detail

KEY

public static final String **KEY**

See Also:
Constant Field Values

Copyright © Oracle Corporation 2009 All Rights Reserved

VALUE

```
public static final String VALUE
```
See Also:
Constant Field Values

TYPE

```
public static final String TYPE
```
See Also:
Constant Field Values

PROPERTIES

```
public static final String[] PROPERTIES
```

PROPERTY

```
public static final CompositeType PROPERTY
```

PROPERTY_TABLE

```
public static final TabularType PROPERTY_TABLE
```

SCALAR_TYPES

```
protected static final java.util.Set<String> SCALAR_TYPES
```

PRIMITIVE_TYPES

```
protected static final java.util.Set<String> PRIMITIVE_TYPES
```

Constructor Detail

OSGiProperties

```
public OSGiProperties()
```

Method Detail

tableFrom

```
public static TabularData tableFrom(java.util.Dictionary properties)
```

Copyright © Oracle Corporation 2009 All Rights Reserved

tableFrom

```
public static TabularData tableFrom(ServiceReference ref)
```

encode

```
public static CompositeData encode(String key,
                                   Object
value)
```

propertiesFrom

```
public static java.util.Hashtable<String,Object>
propertiesFrom(TabularData table)
```

encodeArray

```
protected static CompositeData encodeArray(String key,
Object value,
Class<?> componentClazz)
```

encodeVector

```
protected static CompositeData encodeVector(String key,
java.util.Vector value)
```

typeOf

```
protected static String typeOf(Class<?> clazz)
```

propertyData

```
protected static CompositeData propertyData(String key,
String value,
```

Copyright © Oracle Corporation 2009 All Rights Reserved

```
String type)
```

parse

```
public static Object parse(String value,
                                    String type)
```

parseArray

```
protected static Object parseArray(String value,
                                            java.util.StringTokenize
r tokens)
```

parseScalarArray

```
protected static Object parseScalarArray(String value,
                                                String type)
```

createScalarArray

```
protected static Object[] createScalarArray(String type,
                                                   int size)
```

parsePrimitiveArray

```
protected static Object parsePrimitiveArray(String value,
                                                   String type)
```

parseVector

```
protected static Object parseVector(String value,
                                            java.util.StringTokeniz
er tokens)
```

parseScalar

```
protected static Object parseScalar(String value,
                                            String type)
```

Copyright © Oracle Corporation 2009 All Rights Reserved

5.5.9 OSGiRole

org.osgi.jmx.codec Class OSGiRole

```
Object
  org.osgi.jmx.codec.OSGiRole
```

```
public class OSGiRole
extends Object
```

Field Summary	
protected String	**name**
protected java.util.Hashtable<String,Object>	**properties**
static CompositeType	**ROLE**
protected int	**type**

Constructor Summary
OSGiRole(CompositeData data)
OSGiRole(Role role)

Method Summary	
CompositeData	**asCompositeData**()
String	**getName**()
java.util.Map<String,Object>	**getProperties**()
int	**getType**()

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

Field Detail

name

```
protected String name
```

Copyright © Oracle Corporation 2009 All Rights Reserved

type

```
protected int type
```

properties

```
protected java.util.Hashtable<String,Object> properties
```

ROLE

```
public static final CompositeType ROLE
```

Constructor Detail

OSGiRole

```
public OSGiRole(CompositeData data)
```

OSGiRole

```
public OSGiRole(Role role)
```

Method Detail

asCompositeData

```
public CompositeData asCompositeData()
                                     throws
OpenDataException
```

Throws:
OpenDataException

getName

```
public String getName()
```

Returns:
the name

getType

```
public int getType()
```

Returns:

Copyright © Oracle Corporation 2009 All Rights Reserved

the type

getProperties

```
public java.util.Map<String,Object> getProperties()
```
Returns:
the credentials

5.5.10 OSGiService

org.osgi.jmx.codec Class OSGiService

```
Object
  org.osgi.jmx.codec.OSGiService
```

```
public class OSGiService
extends Object
```

This class represents the CODEC for the composite data representing an OSGi ServiceReference

It serves as both the documentation of the type structure and as the codification of the mechanism to convert to/from the CompositeData.

The structure of the composite data is:

Identifier	String
ObjectClass	Array of String
BundleIdentifier	long
UsingBundles	Array of long

Field Summary

static CompositeType	**SERVICE** The CompositeType representation of the service
static TabularType	**SERVICE_TABLE** The TabularType representation of a list of services

Constructor Summary

OSGiService(CompositeData data) Construct an OSGiService encoded in the CompositeData
OSGiService(long identifier, String[] interfaces, long bundle, long[] usingBundles) Construct an OSGiService

Copyright © Oracle Corporation 2009 All Rights Reserved

OSGiService(ServiceReference reference) Construct an OSGiService from the underlying ServiceReference

Method Summary

CompositeData	**asCompositeData**() Answer the receiver encoded as CompositeData
long	**getBundle**()
long	**getIdentifier**()
String[]	**getInterfaces**()
long[]	**getUsingBundles**()
static TabularData	**tableFrom**(java.util.ArrayList<OSGiService> services) Construct the TabularData representing a list of services

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

Field Detail

SERVICE

public static final CompositeType **SERVICE**
The CompositeType representation of the service

SERVICE_TABLE

public static final TabularType **SERVICE_TABLE**
The TabularType representation of a list of services

Constructor Detail

OSGiService

public **OSGiService**(CompositeData data)
Construct an OSGiService encoded in the CompositeData

Copyright © Oracle Corporation 2009 All Rights Reserved

Parameters:
`data` - - the CompositeData encoding the OSGiService

OSGiService

```
public OSGiService(long identifier,
                   String[] interfaces,
                   long bundle,
                   long[] usingBundles)
```
Construct an OSGiService

Parameters:
`identifier` -
`interfaces` -
`bundle` -
`usingBundles` -

OSGiService

```
public OSGiService(ServiceReference reference)
```
Construct an OSGiService from the underlying ServiceReference

Parameters:
`reference` - - the reference of the service

Method Detail

tableFrom

```
public static TabularData
tableFrom(java.util.ArrayList<OSGiService> services)
```
Construct the TabularData representing a list of services

Parameters:
`services` - - the list of services
Returns:
the TabularData representing the list of OSGiServices

asCompositeData

```
public CompositeData asCompositeData()
```
Answer the receiver encoded as CompositeData

Returns:
the CompositeData encoding of the receiver.

Copyright © Oracle Corporation 2009 All Rights Reserved

getBundle

```
public long getBundle()
```
Returns:
the identifier of the bundle the service belongs to

getIdentifier

```
public long getIdentifier()
```
Returns:
the identifier of the service

getInterfaces

```
public String[] getInterfaces()
```
Returns:
the interfaces implemented by the service

getUsingBundles

```
public long[] getUsingBundles()
```
Returns:
the identifiers of the bundles which are using the service

5.5.11 OSGiServiceEvent

org.osgi.jmx.codec Class OSGiServiceEvent

```
Object
  org.osgi.jmx.codec.OSGiServiceEvent
```

```
public class OSGiServiceEvent
extends Object
```

Author:
Hal Hildebrand Date: Nov 24, 2008 Time: 2:42:48 PM
This class represents the CODEC for the composite data representing a OSGi ServiceEvent

It serves as both the documentation of the type structure and as the codification of the mechanism to convert to/from the CompositeData.

The structure of the composite data is:

Identifier	String
BundleIdentifier	long
BundleLocation	String
ObjectClass	Array of String

Copyright © Oracle Corporation 2009 All Rights Reserved

EventType	int

Field Summary

static CompositeType	**SERVICE_EVENT** The CompositeType representation of the OSGiServiceEvent

Constructor Summary

OSGiServiceEvent(CompositeData data) Construct an OSGiServiceEvent from the CompositeData representing the event
OSGiServiceEvent(long serviceId, long bundleId, String location, String[] interfaces, int eventType) Construct and OSGiServiceEvent
OSGiServiceEvent(ServiceEvent event)

Method Summary

CompositeData	**asCompositeData**() Answer the receiver encoded as CompositeData
long	**getBundleId**()
int	**getEventType**()
String[]	**getInterfaces**()
String	**getLocation**()
long	**getServiceId**()

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

Field Detail

SERVICE_EVENT

public static final CompositeType **SERVICE_EVENT**
The CompositeType representation of the OSGiServiceEvent

Copyright © Oracle Corporation 2009 All Rights Reserved

Constructor Detail

OSGiServiceEvent

```
public OSGiServiceEvent(CompositeData data)
```
Construct an OSGiServiceEvent from the CompositeData representing the event

Parameters:
`data` - = the CompositeData representation of the event

OSGiServiceEvent

```
public OSGiServiceEvent(long serviceId,
                        long bundleId,
                        String location,
                        String[] interfaces,
                        int eventType)
```
Construct and OSGiServiceEvent

Parameters:
`serviceId` -
`bundleId` -
`location` -
`interfaces` -
`eventType` -

OSGiServiceEvent

```
public OSGiServiceEvent(ServiceEvent event)
```

Method Detail

asCompositeData

```
public CompositeData asCompositeData()
```
Answer the receiver encoded as CompositeData

Returns:
the CompositeData encoding of the receiver.

getBundleId

```
public long getBundleId()
```
Returns:
the identifier of the bundle the service belongs to

Copyright © Oracle Corporation 2009 All Rights Reserved

getEventType

```
public int getEventType()
```
Returns:
the type of the event

getInterfaces

```
public String[] getInterfaces()
```
Returns:
the interfaces the service implements

getLocation

```
public String getLocation()
```
Returns:
the location of the bundle the service belongs to

getServiceId

```
public long getServiceId()
```
Returns:
the identifier of the service

5.5.12 OSGiUser

org.osgi.jmx.codec Class OSGiUser

```
Object
  org.osgi.jmx.codec.OSGiUser
```

```
public class OSGiUser
extends Object
```

Field Summary	
protected java.util.Hashtable<String,Object>	**credentials**
protected OSGiRole	**role**
static CompositeType	**USER**

Constructor Summary
OSGiUser(CompositeData data)

Copyright © Oracle Corporation 2009 All Rights Reserved

OSGiUser(User user)

Method Summary

CompositeData	**asCompositeData**()
java.util.Map<String,Object>	**getCredentials**()
OSGiRole	**getRole**()

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

Field Detail

role

```
protected OSGiRole role
```

credentials

```
protected java.util.Hashtable<String,Object> credentials
```

USER

```
public static final CompositeType USER
```

Constructor Detail

OSGiUser

```
public OSGiUser(User user)
```

OSGiUser

```
public OSGiUser(CompositeData data)
```

Method Detail

asCompositeData

Copyright © Oracle Corporation 2009 All Rights Reserved

```
public CompositeData asCompositeData()
                                                                    throws
OpenDataException
```
Throws:
OpenDataException

getRole

```
public OSGiRole getRole()
```
Returns:
the role

getCredentials

```
public java.util.Map<String,Object> getCredentials()
```
Returns:
the credentials

5.5.13 Util

org.osgi.jmx.codec Class Util

```
Object
  org.osgi.jmx.codec.Util
```

```
public class Util
extends Object
```

Author:
Hal Hildebrand Date: Nov 24, 2008 Time: 7:09:25 AM Static utilities used by the system

Field Summary	
static ArrayType	LONG_ARRAY_TYPE
static ArrayType	STRING_ARRAY_TYPE

Constructor Summary
Util()

Method Summary	
static long[]	bundleIds(Bundle[] bundles)

Copyright © Oracle Corporation 2009 All Rights Reserved

static long[]	**bundleIds**(RequiredBundle[] bundles)
static long[]	**getBundleDependencies**(Bundle bundle, PackageAdmin admin)
static String[]	**getBundleExportedPackages**(Bundle b, PackageAdmin admin)
static long[]	**getBundleFragments**(Bundle b, PackageAdmin admin)
static java.util.Map<String,String>	**getBundleHeaders**(Bundle b)
static String[]	**getBundleImportedPackages**(Bundle b, BundleContext bc, PackageAdmin admin)
static long[]	**getBundlesRequiring**(Bundle b, BundleContext bc, PackageAdmin admin)
static String	**getBundleState**(Bundle b)
static RequiredBundle	**getRequiredBundle**(Bundle bundle, BundleContext bc, PackageAdmin admin)
static boolean	**isBundleFragment**(Bundle bundle, PackageAdmin admin)
static boolean	**isBundlePersistentlyStarted**(Bundle bundle, StartLevel sl)
static boolean	**isBundleRequired**(Bundle bundle, BundleContext bc, PackageAdmin admin)
static boolean	**isRequiredBundleRemovalPending**(Bundle bundle, BundleContext bc, PackageAdmin admin)
static long[]	**longArrayFrom**(Long[] array)
static Long[]	**LongArrayFrom**(long[] array)
static String	**packageString**(ExportedPackage pkg)

Copyright © Oracle Corporation 2009 All Rights Reserved

static long[]	serviceIds(ServiceReference[] refs)

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

Field Detail

LONG_ARRAY_TYPE

```
public static ArrayType LONG_ARRAY_TYPE
```

STRING_ARRAY_TYPE

```
public static ArrayType STRING_ARRAY_TYPE
```

Constructor Detail

Util

```
public Util()
```

Method Detail

bundleIds

```
public static long[] bundleIds(Bundle[] bundles)
```

bundleIds

```
public static long[] bundleIds(RequiredBundle[] bundles)
```

getBundlesRequiring

```
public static long[] getBundlesRequiring(Bundle b,
                                         BundleContext bc,
                                         PackageAdmin admin)
```

getBundleExportedPackages

Copyright © Oracle Corporation 2009 All Rights Reserved

```
public static String[] getBundleExportedPackages(Bundle b,
                                                 PackageAdm
in admin)
```

getBundleFragments

```
public static long[] getBundleFragments(Bundle b,
                                        PackageAdmin admin)
```

getBundleHeaders

```
public static java.util.Map<String,String> getBundleHeaders(Bundle b)
```

getBundleImportedPackages

```
public static String[] getBundleImportedPackages(Bundle b,
                                                 BundleCont
ext bc,
                                                 PackageAdm
in admin)
```

getBundleDependencies

```
public static long[] getBundleDependencies(Bundle bundle,
                                           PackageAdmin admin)
```

getBundleState

```
public static String getBundleState(Bundle b)
```

getRequiredBundle

```
public static RequiredBundle getRequiredBundle(Bundle bundle,
BundleContext bc,
PackageAdmin admin)
```

isBundleFragment

Copyright © Oracle Corporation 2009 All Rights Reserved

```
public static boolean isBundleFragment(Bundle bundle,
                                       PackageAdmin admin)
```

isBundlePersistentlyStarted

```
public static boolean isBundlePersistentlyStarted(Bundle bundle,
                                                  StartLevel sl)
```

isBundleRequired

```
public static boolean isBundleRequired(Bundle bundle,
                                       BundleContext bc,
                                       PackageAdmin admin)
```

isRequiredBundleRemovalPending

```
public static boolean isRequiredBundleRemovalPending(Bundle bundle,
                                                     BundleContext bc,
                                                     PackageAdmin adm
in)
```

packageString

```
public static String packageString(ExportedPackage pkg)
```

serviceIds

```
public static long[] serviceIds(ServiceReference[] refs)
```

LongArrayFrom

```
public static Long[] LongArrayFrom(long[] array)
```

longArrayFrom

```
public static long[] longArrayFrom(Long[] array)
```

Copyright © Oracle Corporation 2009 All Rights Reserved

5.6 Core Command and Control Interfaces

These interfaces provide the management client with batch operations on the core framework apis as well as access to the core OSGi artifacts, Bundles, Services and Packages.

5.6.1 Interface FrameworkMBean

org.osgi.jmx.core Interface FrameworkMBean

```
public interface FrameworkMBean
```

The FrameworkMbean provides mechanisms to exert control over the framework. For many operations, it provides a batch mechanism to avoid excessive message passing when interacting remotely.

Field Summary	
static String[]	**BUNDLE_ACTION_RESULT** The item names in the CompositeData representing the result of a batch operation
static String	**BUNDLE_BATCH_ACTION_RESULT** The name of the CompositeType which represents the result of a batch operation
static String	**BUNDLE_BATCH_INSTALL_RESULT** The name of the CompositeType which represents the result of a batch install operation
static String	**BUNDLE_COMPLETED** The name of the item containing the list of bundles completing the batch operation in the CompositeData
static String	**BUNDLE_ERROR_MESSAGE** The name of the item containing the error message of the batch operation in the CompositeData
static String	**BUNDLE_IN_ERROR** The name of the item containing the bundle which caused the error during the batch operation in the CompositeData
static String	**BUNDLE_REMAINING** The name of the item containing the list of remaing bundles unproccessed by the failing batch operation in the CompositeData
static String	**BUNDLE_SUCCESS**

Copyright © Oracle Corporation 2009 All Rights Reserved

	The name of the item containing the success status of the batch operation in the CompositeData

Method Summary

int	**getFrameworkStartLevel**() Retrieve the framework start level
int	**getInitialBundleStartLevel**() Answer the initial start level assigned to a bundle when it is first started
long	**installBundle**(String location) Install the bundle indicated by the bundleLocations
long	**installBundle**(String location, String url) Install the bundle indicated by the bundleLocations
CompositeData	**installBundles**(String[] locations) Batch install the bundles indicated by the list of bundleLocationUrls
CompositeData	**installBundles**(String[] locations, String[] urls) Batch install the bundles indicated by the list of bundleLocationUrls
void	**refreshPackages**(long bundleIdentifier) Force the update, replacement or removal of the pacakges identified by the list of bundles
CompositeData	**refreshPackages**(long[] bundleIdentifiers) Force the update, replacement or removal of the pacakges identified by the list of bundles
boolean	**resolveBundle**(long bundleIdentifier) Resolve the bundle indicated by the unique symbolic name and version
boolean	**resolveBundles**(long[] bundleIdentifiers) Batch resolve the bundles indicated by the list of bundle identifiers
void	**restartFramework**() Restart the framework by updating the system bundle
void	**setBundleStartLevel**(long bundleIdentifier, int newlevel) Set the start level for the bundle

Copyright © Oracle Corporation 2009 All Rights Reserved

	identifier
CompositeData	**setBundleStartLevels**(long[] bundleIdentifiers, int[] newlevels) Set the start levels for the list of bundles
void	**setFrameworkStartLevel**(int newlevel) Set the start level for the framework
void	**setInitialBundleStartLevel**(int newlevel) Set the initial start level assigned to a bundle when it is first started
void	**shutdownFramework**() Shutdown the framework by stopping the system bundle
void	**startBundle**(long bundleIdentifier) Start the bundle indicated by the bundle identifier
CompositeData	**startBundles**(long[] bundleIdentifiers) Batch start the bundles indicated by the list of bundle identifier
void	**stopBundle**(long bundleIdentifier) Stop the bundle indicated by the bundle identifier
CompositeData	**stopBundles**(long[] bundleIdentifiers) Batch stop the bundles indicated by the list of bundle identifier
void	**uninstallBundle**(long bundleIdentifier) Uninstall the bundle indicated by the bundle identifier
CompositeData	**uninstallBundles**(long[] bundleIdentifiers) Batch uninstall the bundles indicated by the list of bundle identifiers
void	**updateBundle**(long bundleIdentifier) Update the bundle indicated by the bundle identifier
void	**updateBundle**(long bundleIdentifier, String url) Update the bundle identified by the bundle identifier
CompositeData	**updateBundles**(long[] bundleIdentifiers) Batch update the bundles indicated

Copyright © Oracle Corporation 2009 All Rights Reserved

	by the list of bundle identifier
CompositeData	**updateBundles**(long[] bundleIdentifiers, String[] urls) Update the bundle uniquely identified by the bundle symbolic name and version using the contents of the supplied urls
void	**updateFramework**() Update the framework by updating the system bundle

Field Detail

BUNDLE_COMPLETED

static final String **BUNDLE_COMPLETED**
The name of the item containing the list of bundles completing the batch operation in the CompositeData

See Also:
Constant Field Values

BUNDLE_ERROR_MESSAGE

static final String **BUNDLE_ERROR_MESSAGE**
The name of the item containing the error message of the batch operation in the CompositeData

See Also:
Constant Field Values

BUNDLE_IN_ERROR

static final String **BUNDLE_IN_ERROR**
The name of the item containing the bundle which caused the error during the batch operation in the CompositeData

See Also:
Constant Field Values

BUNDLE_REMAINING

static final String **BUNDLE_REMAINING**
The name of the item containing the list of remaing bundles unproccessed by the failing batch operation in the CompositeData

Copyright © Oracle Corporation 2009 All Rights Reserved

See Also:
Constant Field Values

BUNDLE_SUCCESS

`static final String` **`BUNDLE_SUCCESS`**
The name of the item containing the success status of the batch operation in the CompositeData

See Also:
Constant Field Values

BUNDLE_ACTION_RESULT

`static final String[]` **`BUNDLE_ACTION_RESULT`**
The item names in the CompositeData representing the result of a batch operation

BUNDLE_BATCH_ACTION_RESULT

`static final String` **`BUNDLE_BATCH_ACTION_RESULT`**
The name of the CompositeType which represents the result of a batch operation

See Also:
Constant Field Values

BUNDLE_BATCH_INSTALL_RESULT

`static final String` **`BUNDLE_BATCH_INSTALL_RESULT`**
The name of the CompositeType which represents the result of a batch install operation

See Also:
Constant Field Values

Method Detail

getFrameworkStartLevel

```
int getFrameworkStartLevel()
                           throws IOException
```

Retrieve the framework start level

Returns:

Copyright © Oracle Corporation 2009 All Rights Reserved

the framework start level
Throws:
`IOException` - if the operation failed

getInitialBundleStartLevel

```
int getInitialBundleStartLevel()
                                    throws IOException
```

Answer the initial start level assigned to a bundle when it is first started

Returns:
the start level
Throws:
`IOException` - if the operation failed

installBundle

```
long installBundle(String location)
                   throws IOException
```

Install the bundle indicated by the bundleLocations

Parameters:
`location` - - the location of the bundle to install
Returns:
the bundle id the installed bundle
Throws:
`IOException` - if the operation does not succeed

installBundle

```
long installBundle(String location,
                   String url)
                   throws IOException
```

Install the bundle indicated by the bundleLocations

Parameters:
`location` - - the location to assign to the bundle
`url` - - the URL which will supply the bytes for the bundle
Returns:
the bundle id the installed bundle
Throws:
`IOException` - if the operation does not succeed

installBundles

```
CompositeData installBundles(String[] locations)
```

Copyright © Oracle Corporation 2009 All Rights Reserved

throws IOException
Batch install the bundles indicated by the list of bundleLocationUrls

Parameters:
locations - - the array of locations of the bundles to install
Returns:
the resulting state from executing the operation
Throws:
IOException - if the operation does not succeed
See Also:
BatchBundleResult for the precise specification of the CompositeData type representing the returned result.

installBundles

```
CompositeData installBundles(String[] locations,
                             String[] urls)
                             throws IOException
```
Batch install the bundles indicated by the list of bundleLocationUrls

Parameters:
locations - - the array of locations to assign to the installed bundles
urls - - the array of urls which supply the bundle bytes
Returns:
the resulting state from executing the operation
Throws:
IOException - if the operation does not succeed
See Also:
BatchBundleResult for the precise specification of the CompositeData type representing the returned result.

refreshPackages

```
void refreshPackages(long bundleIdentifier)
                     throws IOException
```
Force the update, replacement or removal of the pacakges identified by the list of bundles

Copyright © Oracle Corporation 2009 All Rights Reserved

Parameters:
`bundleIdentifier` - - the bundle identifier
Throws:
`IOException` - if the operation failed
See Also:
`BundleBatchActionResult for the precise specification of the CompositeData type representing the returned result.`

refreshPackages

```
CompositeData refreshPackages(long[] bundleIdentifiers)
                                                        throws
IOException
```
Force the update, replacement or removal of the pacakges identified by the list of bundles

Parameters:
`bundleIdentifiers` - - the array of bundle identifiers
Returns:
the resulting state from executing the operation
Throws:
`IOException` - if the operation failed
See Also:
`BundleBatchActionResult for the precise specification of the CompositeData type representing the returned result.`

resolveBundle

```
boolean resolveBundle(long bundleIdentifier)
                      throws IOException
```
Resolve the bundle indicated by the unique symbolic name and version

Parameters:
`bundleIdentifier` - - the bundle identifier
Returns:
true if the bundle was resolved, false otherwise
Throws:
`IOException` - if the operation does not succeed
`IllegalArgumentException` - if the bundle indicated does not exist

resolveBundles

Copyright © Oracle Corporation 2009 All Rights Reserved

```
boolean resolveBundles(long[] bundleIdentifiers)
                       throws IOException
```
Batch resolve the bundles indicated by the list of bundle identifiers

Parameters:
`bundleIdentifiers` - = the identifiers of the bundles to resolve
Returns:
true if the bundles were resolved, false otherwise
Throws:
`IOException` - if the operation does not succeed

restartFramework

```
void restartFramework()
                      throws IOException
```
Restart the framework by updating the system bundle

Throws:
`IOException` - if the operation failed

setBundleStartLevel

```
void setBundleStartLevel(long bundleIdentifier,
                         int newlevel)
                         throws IOException
```
Set the start level for the bundle identifier

Parameters:
`bundleIdentifier` - - the bundle identifier
`newlevel` - - the new start level for the bundle
Throws:
`IOException` - if the operation failed

setBundleStartLevels

```
CompositeData setBundleStartLevels(long[] bundleIdentifiers,
                                                                  int[] n
ewlevels)
                                                                  throws
IOException
```
Set the start levels for the list of bundles

Parameters:

Copyright © Oracle Corporation 2009 All Rights Reserved

`bundleIdentifiers` - - the array of bundle identifiers
`newlevels` - - the array of new start level for the bundles
Returns:
the resulting state from executing the operation
Throws:
`IOException` - if the operation failed
See Also:
`BundleBatchActionResult for the precise specification of the CompositeData type representing the returned result.`

setFrameworkStartLevel

```
void setFrameworkStartLevel(int newlevel)
                            throws IOException
```

Set the start level for the framework

Parameters:
`newlevel` - - the new start level
Throws:
`IOException` - if the operation failed

setInitialBundleStartLevel

```
void setInitialBundleStartLevel(int newlevel)
                                throws IOException
```

Set the initial start level assigned to a bundle when it is first started

Parameters:
`newlevel` - - the new start level
Throws:
`IOException` - if the operation failed

shutdownFramework

```
void shutdownFramework()
                       throws IOException
```

Shutdown the framework by stopping the system bundle

Throws:
`IOException` - if the operation failed

startBundle

```
void startBundle(long bundleIdentifier)
```

Copyright © Oracle Corporation 2009 All Rights Reserved

```
                    throws IOException
```
Start the bundle indicated by the bundle identifier

Parameters:
`bundleIdentifier` - - the bundle identifier
Throws:
`IOException` - if the operation does not succeed
`IllegalArgumentException` - if the bundle indicated does not exist

startBundles

```
CompositeData startBundles(long[] bundleIdentifiers)
                                                    throws
IOException
```
Batch start the bundles indicated by the list of bundle identifier

Parameters:
`bundleIdentifiers` - - the array of bundle identifiers
Returns:
the resulting state from executing the operation
Throws:
`IOException` - if the operation does not succeed
See Also:
`BundleBatchActionResult for the precise specification of the CompositeData type representing the returned result.`

stopBundle

```
void stopBundle(long bundleIdentifier)
                throws IOException
```
Stop the bundle indicated by the bundle identifier

Parameters:
`bundleIdentifier` - - the bundle identifier
Throws:
`IOException` - if the operation does not succeed
`IllegalArgumentException` - if the bundle indicated does not exist

stopBundles

```
CompositeData stopBundles(long[] bundleIdentifiers)
```

Copyright © Oracle Corporation 2009 All Rights Reserved

throws IOException
Batch stop the bundles indicated by the list of bundle identifier

Parameters:
bundleIdentifiers - - the array of bundle identifiers
Returns:
the resulting state from executing the operation
Throws:
IOException - if the operation does not succeed
See Also:
BundleBatchActionResult for the precise specification of the CompositeData type representing the returned result.

uninstallBundle

```
void uninstallBundle(long bundleIdentifier)
                     throws IOException
```
Uninstall the bundle indicated by the bundle identifier

Parameters:
bundleIdentifier - - the bundle identifier
Throws:
IOException - if the operation does not succeed
IllegalArgumentException - if the bundle indicated does not exist

uninstallBundles

```
CompositeData uninstallBundles(long[] bundleIdentifiers)
                                                        throws
IOException
```
Batch uninstall the bundles indicated by the list of bundle identifiers

Parameters:
bundleIdentifiers - - the array of bundle identifiers
Returns:
the resulting state from executing the operation
Throws:
IOException - if the operation does not succeed
See Also:

Copyright © Oracle Corporation 2009 All Rights Reserved

BundleBatchActionResult for the precise specification of the CompositeData type representing the returned result.

updateBundle

```
void updateBundle(long bundleIdentifier)
                  throws IOException
```
Update the bundle indicated by the bundle identifier

Parameters:
`bundleIdentifier` - - the bundle identifier
Throws:
`IOException` - if the operation does not succeed
`IllegalArgumentException` - if the bundle indicated does not exist

updateBundle

```
void updateBundle(long bundleIdentifier,
                  String url)
                  throws IOException
```
Update the bundle identified by the bundle identifier

Parameters:
`bundleIdentifier` - - the bundle identifier
`url` - - the URL to use to update the bundle
Throws:
`IOException` - if the operation does not succeed
`IllegalArgumentException` - if the bundle indicated does not exist

updateBundles

```
CompositeData updateBundles(long[] bundleIdentifiers)
                                                    throws
IOException
```
Batch update the bundles indicated by the list of bundle identifier

Parameters:
`bundleIdentifiers` - - the array of bundle identifiers
Returns:
the resulting state from executing the operation
Throws:
`IOException` - if the operation does not succeed

Copyright © Oracle Corporation 2009 All Rights Reserved

See Also:
BundleBatchActionResult for the precise specification of the CompositeData type representing the returned result.

updateBundles

```
CompositeData updateBundles(long[] bundleIdentifiers,
                            String[] urls)
                            throws
IOException
```

Update the bundle uniquely identified by the bundle symbolic name and version using the contents of the supplied urls

Parameters:
`bundleIdentifiers` - - the array of bundle identifiers
`urls` - - the array of URLs to use to update the bundles
Returns:
the resulting state from executing the operation
Throws:
`IOException` - if the operation does not succeed
`IllegalArgumentException` - if the bundle indicated does not exist
See Also:
BundleBatchActionResult for the precise specification of the CompositeData type representing the returned result.

updateFramework

```
void updateFramework()
                     throws IOException
```

Update the framework by updating the system bundle

Throws:
`IOException` - if the operation failed

Copyright © Oracle Corporation 2009 All Rights Reserved

5.6.2 Interface BundleStateMBean

org.osgi.jmx.core Interface BundleStateMBean

`public interface BundleStateMBean`

This MBean represents the Bundle state of the framework. This MBean also emits events that clients can use to get notified of the changes in the bundle state of the framework.

See OSGiBundleEvent for the precise definition of the CompositeData for the notification sent.

Field Summary	
static String[]	**BUNDLE** The item names in the CompositeData representing an OSGi Bundle
static String[]	**BUNDLE_EVENT** The item names in the CompositeData representing the event raised for bundle events within the OSGi container by this bean
static String	**BUNDLE_EVENT_TYPE** The type of the event which is emitted when bundle state changes occur in the OSGi container
static String	**BUNDLE_EXPORTED_PACKAGES** The name of the item containing the exported packages in the CompositeData
static String	**BUNDLE_FRAGMENT** The name of the item containing the fragment status in the CompositeData
static String	**BUNDLE_FRAGMENTS** The name of the item containing the list of fragments the bundle is host to in the CompositeData representing a Bundle
static String	**BUNDLE_HEADER_TYPE**
static String	**BUNDLE_HEADERS** The name of the item containing the bundle headers in the CompositeData
static String	**BUNDLE_HEADERS_TYPE**
static String	**BUNDLE_HOSTS** The name of the item containing the bundle identifiers representing the hosts

Copyright © Oracle Corporation 2009 All Rights Reserved

static String	**BUNDLE_ID** The name of the item containing the bundle identifier in the CompositeData
static String	**BUNDLE_IMPORTED_PACKAGES** The name of the item containing the imported packages in the CompositeData
static String	**BUNDLE_LAST_MODIFIED** The name of the item containing the last modified time in the CompositeData
static String	**BUNDLE_LOCATION** The name of the item containing the bundle location in the CompositeData
static String	**BUNDLE_PERSISTENTLY_STARTED** The name of the item containing the indication of persistently started in the CompositeData
static String	**BUNDLE_REGISTERED_SERVICES** The name of the item containing the registered services of the bundle in the CompositeData
static String	**BUNDLE_REMOVAL_PENDING** The name of the item containing the indication of removal pending in the CompositeData
static String	**BUNDLE_REQUIRED** The name of the item containing the required status in the CompositeData
static String	**BUNDLE_REQUIRED_BUNDLES** The name of the item containing the required bundles in the CompositeData
static String	**BUNDLE_REQUIRING_BUNDLES** The name of the item containing the bundles requiring this bundle in the CompositeData
static String	**BUNDLE_SERVICES_IN_USE** The name of the item containing the services in use by this bundle in the CompositeData
static String	**BUNDLE_START_LEVEL** The name of the item containing the start level in the CompositeData
static String	**BUNDLE_STATE** The name of the item containing the bundle state in the CompositeData
static String	**BUNDLE_SYMBOLIC_NAME** The name of the item containing the symbolic name in the CompositeData
static String	**BUNDLE_TYPE_NAME** The name CompositeData type for a bundle
static String	**EVENT_TYPE**

Copyright © Oracle Corporation 2009 All Rights Reserved

	The name of the item containing the event type in the CompositeData

Method Summary	
TabularData	**getBundles**() Answer the bundle state of the system in tabular form Each row of the returned table represents a single bundle.
long[]	**getDependencies**(long bundleIdentifier) Answer the list of identifiers of the bundles this bundle depends upon
String[]	**getExportedPackages**(long bundleId) Answer the list of exported packages for this bundle
long[]	**getFragments**(long bundleId) Answer the list of the bundle ids of the fragments associated with this bundle
TabularData	**getHeaders**(long bundleId) Answer the headers for the bundle uniquely identified by the bundle id
long[]	**getHosts**(long fragment) Answer the list of bundle ids of the bundles which host a fragment
String[]	**getImportedPackages**(long bundleId) Answer the array of the packages imported by this bundle
long	**getLastModified**(long bundleId) Answer the last modified time of a bundle
long[]	**getRegisteredServices**(long bundleId) Answer the list of service identifiers representing the services this bundle exports
long[]	**getRequiringBundles**(long bundleIdentifier) Answer the list of identifiers of the bundles which require this bundle
long[]	**getServicesInUse**(long bundleIdentifier) Answer the list of service identifiers which refer to the the services this bundle is using
int	**getStartLevel**(long bundleId) Answer the start level of the bundle
String	**getState**(long bundleId)

Copyright © Oracle Corporation 2009 All Rights Reserved

	Answer the symbolic name of the state of the bundle
String	**getSymbolicName**(long bundleId) Answer the symbolic name of the bundle
boolean	**isFragment**(long bundleId) Answer whether the bundle is a fragment or not
boolean	**isPersistentlyStarted**(long bundleId) Answer if the bundle is persistently started when its start level is reached
boolean	**isRemovalPending**(long bundleId) Answer true if the bundle is pending removal
boolean	**isRequired**(long bundleId) Answer true if the bundle is required by another bundle

Field Detail

BUNDLE_EVENT_TYPE

`static final String` **BUNDLE_EVENT_TYPE**
The type of the event which is emitted when bundle state changes occur in the OSGi container

See Also:
Constant Field Values

BUNDLE_EXPORTED_PACKAGES

`static final String` **BUNDLE_EXPORTED_PACKAGES**
The name of the item containing the exported packages in the CompositeData

See Also:
Constant Field Values

BUNDLE_FRAGMENT

`static final String` **BUNDLE_FRAGMENT**
The name of the item containing the fragment status in the CompositeData

See Also:

Copyright © Oracle Corporation 2009 All Rights Reserved

Constant Field Values

BUNDLE_FRAGMENTS

`static final String` **`BUNDLE_FRAGMENTS`**
The name of the item containing the list of fragments the bundle is host to in the CompositeData representing a Bundle

See Also:
Constant Field Values

BUNDLE_HEADERS

`static final String` **`BUNDLE_HEADERS`**
The name of the item containing the bundle headers in the CompositeData

See Also:
Constant Field Values

BUNDLE_HOSTS

`static final String` **`BUNDLE_HOSTS`**
The name of the item containing the bundle identifiers representing the hosts

See Also:
Constant Field Values

BUNDLE_ID

`static final String` **`BUNDLE_ID`**
The name of the item containing the bundle identifier in the CompositeData

See Also:
Constant Field Values

BUNDLE_IMPORTED_PACKAGES

`static final String` **`BUNDLE_IMPORTED_PACKAGES`**
The name of the item containing the imported packages in the CompositeData

See Also:
Constant Field Values

Copyright © Oracle Corporation 2009 All Rights Reserved

BUNDLE_LAST_MODIFIED

`static final String` **`BUNDLE_LAST_MODIFIED`**
The name of the item containing the last modified time in the CompositeData

See Also:
Constant Field Values

BUNDLE_LOCATION

`static final String` **`BUNDLE_LOCATION`**
The name of the item containing the bundle location in the CompositeData

See Also:
Constant Field Values

BUNDLE_PERSISTENTLY_STARTED

`static final String` **`BUNDLE_PERSISTENTLY_STARTED`**
The name of the item containing the indication of persistently started in the CompositeData

See Also:
Constant Field Values

BUNDLE_REGISTERED_SERVICES

`static final String` **`BUNDLE_REGISTERED_SERVICES`**
The name of the item containing the registered services of the bundle in the CompositeData

See Also:
Constant Field Values

BUNDLE_REMOVAL_PENDING

`static final String` **`BUNDLE_REMOVAL_PENDING`**
The name of the item containing the indication of removal pending in the CompositeData

See Also:
Constant Field Values

BUNDLE_REQUIRED

`static final String` **`BUNDLE_REQUIRED`**

Copyright © Oracle Corporation 2009 All Rights Reserved

The name of the item containing the required status in the CompositeData

See Also:
Constant Field Values

BUNDLE_REQUIRED_BUNDLES

`static final String` **`BUNDLE_REQUIRED_BUNDLES`**
The name of the item containing the required bundles in the CompositeData

See Also:
Constant Field Values

BUNDLE_REQUIRING_BUNDLES

`static final String` **`BUNDLE_REQUIRING_BUNDLES`**
The name of the item containing the bundles requiring this bundle in the CompositeData

See Also:
Constant Field Values

BUNDLE_SERVICES_IN_USE

`static final String` **`BUNDLE_SERVICES_IN_USE`**
The name of the item containing the services in use by this bundle in the CompositeData

See Also:
Constant Field Values

BUNDLE_START_LEVEL

`static final String` **`BUNDLE_START_LEVEL`**
The name of the item containing the start level in the CompositeData

See Also:
Constant Field Values

BUNDLE_STATE

`static final String` **`BUNDLE_STATE`**
The name of the item containing the bundle state in the CompositeData

Copyright © Oracle Corporation 2009 All Rights Reserved

See Also:
Constant Field Values

BUNDLE_SYMBOLIC_NAME

`static final String` **`BUNDLE_SYMBOLIC_NAME`**
The name of the item containing the symbolic name in the CompositeData

See Also:
Constant Field Values

BUNDLE_TYPE_NAME

`static final String` **`BUNDLE_TYPE_NAME`**
The name CompositeData type for a bundle

See Also:
Constant Field Values

EVENT_TYPE

`static final String` **`EVENT_TYPE`**
The name of the item containing the event type in the CompositeData

See Also:
Constant Field Values

BUNDLE

`static final String[]` **`BUNDLE`**
The item names in the CompositeData representing an OSGi Bundle

BUNDLE_EVENT

`static final String[]` **`BUNDLE_EVENT`**
The item names in the CompositeData representing the event raised for bundle events within the OSGi container by this bean

Copyright © Oracle Corporation 2009 All Rights Reserved

BUNDLE_HEADERS_TYPE

```
static final String BUNDLE_HEADERS_TYPE
```
See Also:
Constant Field Values

BUNDLE_HEADER_TYPE

```
static final String BUNDLE_HEADER_TYPE
```
See Also:
Constant Field Values

Method Detail

getDependencies

```
long[] getDependencies(long bundleIdentifier)
                       throws IOException
```
Answer the list of identifiers of the bundles this bundle depends upon

Parameters:
`bundleIdentifier` - - the bundle identifier
Returns:
the list of bundle identifiers
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the bundle indicated does not exist

getBundles

```
TabularData getBundles()
                                                    throws IOException
```
Answer the bundle state of the system in tabular form Each row of the returned table represents a single bundle. For each bundle, the following row is returned
location - String
bundle identifier - String
symbolic name - String
start level - int
state - String
last modified - long
persistently started - boolean
removal pending - boolean
required - boolean
fragement - boolean
registered services - long[]
services in use - long[]
headers - TabularData
exported packages - String[]
imported packages - String[]
fragments - long[]

Copyright © Oracle Corporation 2009 All Rights Reserved

hosts - long[]
required bundles - long[]
requiring bundles - long[]

Returns:
the tabular respresentation of the bundle state
Throws:
`IOException`
See Also:
[for the precise specifiction of the CompositeType definition for each row of the table.]

getExportedPackages

```
String[] getExportedPackages(long bundleId)
                                        throws IOException
```

Answer the list of exported packages for this bundle

Parameters:
`bundleId` -
Returns:
the array of package names, combined with their version in the format
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the bundle indicated does not exist

getFragments

```
long[] getFragments(long bundleId)
                    throws IOException
```

Answer the list of the bundle ids of the fragments associated with this bundle

Parameters:
`bundleId` -
Returns:
the array of bundle identifiers
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the bundle indicated does not exist

getHeaders

```
TabularData getHeaders(long bundleId)
                                                throws IOException
```

Answer the headers for the bundle uniquely identified by the bundle id

Copyright © Oracle Corporation 2009 All Rights Reserved

Parameters:
`bundleId` - - the unique identifer of the bundle
Returns:
the table of associated header key and values
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the bundle indicated does not exist
See Also:
`for the precise specifiction of the TabularType`

getHosts

```
long[] getHosts(long fragment)
                throws IOException
```

Answer the list of bundle ids of the bundles which host a fragment

Parameters:
`fragment` - - the bundle id of the fragment
Returns:
the array of bundle identifiers
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the bundle indicated does not exist

getImportedPackages

```
String[] getImportedPackages(long bundleId)
                                          throws IOException
```

Answer the array of the packages imported by this bundle

Parameters:
`bundleId` - - the bundle identifier
Returns:
the array of package names, combined with their version in the format
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the bundle indicated does not exist

getLastModified

```
long getLastModified(long bundleId)
                     throws IOException
```

Answer the last modified time of a bundle

Parameters:
`bundleId` - - the unique identifier of a bundle

Copyright © Oracle Corporation 2009 All Rights Reserved

Returns:
the last modified time
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the bundle indicated does not exist

getRegisteredServices

```
long[] getRegisteredServices(long bundleId)
                                    throws IOException
```
Answer the list of service identifiers representing the services this bundle exports

Parameters:
`bundleId` - - the bundle identifier
Returns:
the list of service identifiers
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the bundle indicated does not exist

getRequiringBundles

```
long[] getRequiringBundles(long bundleIdentifier)
                                  throws IOException
```
Answer the list of identifiers of the bundles which require this bundle

Parameters:
`bundleIdentifier` - - the bundle identifier
Returns:
the list of bundle identifiers
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the bundle indicated does not exist

getServicesInUse

```
long[] getServicesInUse(long bundleIdentifier)
                                throws IOException
```
Answer the list of service identifiers which refer to the the services this bundle is using

Parameters:
`bundleIdentifier` - - the bundle identifier
Returns:
the list of service identifiers
Throws:
`IOException` - if the operation fails

Copyright © Oracle Corporation 2009 All Rights Reserved

`IllegalArgumentException` - if the bundle indicated does not exist

getStartLevel

```
int getStartLevel(long bundleId)
                  throws IOException
```

Answer the start level of the bundle

Parameters:
`bundleId` - - the identifier of the bundle
Returns:
the start level
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the bundle indicated does not exist

getState

```
String getState(long bundleId)
                            throws IOException
```

Answer the symbolic name of the state of the bundle

Parameters:
`bundleId` - - the identifier of the bundle
Returns:
the string name of the bundle state
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the bundle indicated does not exist

getSymbolicName

```
String getSymbolicName(long bundleId)
                                     throws IOException
```

Answer the symbolic name of the bundle

Parameters:
`bundleId` - - the identifier of the bundle
Returns:
the symbolic name
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the bundle indicated does not exist

isPersistentlyStarted

Copyright © Oracle Corporation 2009 All Rights Reserved

```
boolean isPersistentlyStarted(long bundleId)
                                throws IOException
```
Answer if the bundle is persistently started when its start level is reached

Parameters:
`bundleId` - - the identifier of the bundle
Returns:
true if the bundle is persistently started
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the bundle indicated does not exist

isFragment

```
boolean isFragment(long bundleId)
                   throws IOException
```
Answer whether the bundle is a fragment or not

Parameters:
`bundleId` - - the identifier of the bundle
Returns:
true if the bundle is a fragment
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the bundle indicated does not exist

isRemovalPending

```
boolean isRemovalPending(long bundleId)
                         throws IOException
```
Answer true if the bundle is pending removal

Parameters:
`bundleId` - - the identifier of the bundle
Returns:
true if the bundle is pending removal
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the bundle indicated does not exist

isRequired

```
boolean isRequired(long bundleId)
                   throws IOException
```
Answer true if the bundle is required by another bundle

Copyright © Oracle Corporation 2009 All Rights Reserved

Parameters:
`bundleId` - - the identifier of the bundle
Returns:
true if the bundle is required by another bundle
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the bundle indicated does not exist

Copyright © Oracle Corporation 2009 All Rights Reserved

5.6.3 Interface PackageStateMBean

org.osgi.jmx.core Interface PackageStateMBean

`public interface PackageStateMBean`

Author:
Hal Hildebrand Date: Sep 23, 2008 Time: 9:04:23 AM
This MBean represents the Package state of the framework.

Field Summary	
static String	**BUNDLE_IDENTIFIER** The name of the item containing the bundle identifier in the CompositeData
static String	**IMPORTING_BUNDLES** The name of the item containing the importing bundles in the CompositeData
static String[]	**PACKAGE** The item names in the CompositeData representing the OSGi Package
static String	**PACKAGE_NAME** The name of the item containing the package name in the CompositeData
static String	**PACKAGE_PENDING_REMOVAL** The name of the item containing the pending removal status of the package in the CompositeData
static String	**PACKAGE_VERSION** The name of the item containing the package version in the CompositeData

Method Summary	
long	**getExportingBundle**(String packageName, String version) Answer the identifier of the bundle exporting the package
long[]	**getImportingBundles**(String packageName, String version) Answer the list of identifiers of the bundles importing the package
TabularData	**getPackages**() Answer the package state of the system in tabular form

Copyright © Oracle Corporation 2009 All Rights Reserved

boolean	**isRemovalPending**(String packageName, String version) Answer if this package is exported by a bundle which has been updated or uninstalled

Field Detail

BUNDLE_IDENTIFIER

static final String **BUNDLE_IDENTIFIER**
The name of the item containing the bundle identifier in the CompositeData

See Also:
Constant Field Values

IMPORTING_BUNDLES

static final String **IMPORTING_BUNDLES**
The name of the item containing the importing bundles in the CompositeData

See Also:
Constant Field Values

PACKAGE_NAME

static final String **PACKAGE_NAME**
The name of the item containing the package name in the CompositeData

See Also:
Constant Field Values

PACKAGE_PENDING_REMOVAL

static final String **PACKAGE_PENDING_REMOVAL**
The name of the item containing the pending removal status of the package in the CompositeData

See Also:
Constant Field Values

PACKAGE_VERSION

static final String **PACKAGE_VERSION**

Copyright © Oracle Corporation 2009 All Rights Reserved

The name of the item containing the package version in the CompositeData

See Also:
Constant Field Values

PACKAGE

```
static final String[] PACKAGE
```
The item names in the CompositeData representing the OSGi Package

Method Detail

getExportingBundle

```
long getExportingBundle(String packageName,
                        String version)
                        throws IOException
```
Answer the identifier of the bundle exporting the package

Parameters:
`packageName` - - the package name
`version` - - the version of the package
Returns:
the bundle identifier or -1 if there is no bundle
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the package indicated does not exist

getImportingBundles

```
long[] getImportingBundles(String packageName,
                           String version)
                           throws IOException
```
Answer the list of identifiers of the bundles importing the package

Parameters:
`packageName` - - the package name
`version` - - the version of the package
Returns:
the list of bundle identifiers
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the package indicated does not exist

getPackages

Copyright © Oracle Corporation 2009 All Rights Reserved

`TabularData` **`getPackages()`**
Answer the package state of the system in tabular form

Returns:
the tabular respresentation of the package state
Throws:
`IOException`
See Also:
for the details of the TabularType.
Each row of the returned table represents a single package. For each package, the following row is returned
- name - String
- version - String
- removal pending - boolean
- exporting bundle - long
- importing bundles - long[]

isRemovalPending

```
boolean isRemovalPending(String packageName,
                         String version)
                         throws IOException
```
Answer if this package is exported by a bundle which has been updated or uninstalled

Parameters:
`packageName` - - the package name
`version` - - the version of the package
Returns:
true if this package is being exported by a bundle that has been updated or uninstalled.
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the package indicated does not exist

5.6.4 Interface ServiceStateMBean

org.osgi.jmx.core **Interface ServiceStateMBean**

```
public interface ServiceStateMBean
```

Author:
Hal Hildebrand Date: Sep 23, 2008 Time: 8:57:33 AM
This MBean represents the Service state of the framework. This MBean also emits events that clients can use to get notified of the changes in the service state of the framework.

See OSGiBundleEvent for the precise definition of the CompositeData for the notification sent.

Copyright © Oracle Corporation 2009 All Rights Reserved

Field Summary	
static String	**BUNDLE_IDENTIFIER** The name of the item containing the bundle identifier in the CompositeData
static String	**BUNDLE_LOCATION** The name of the item containing the bundle location in the CompositeData
static String	**EVENT_TYPE** The name of the item containing the event type in the CompositeData
static String	**OBJECT_CLASS** The name of the item containing the interfaces of the service in the CompositeData
static String[]	**SERVICE** The item names in the CompositeData representing the service
static String[]	**SERVICE_EVENT** The item names in the CompositeData representing the ServiceEvent
static String	**SERVICE_EVENT_TYPE** The type of the JMX event raised in response to ServiceEvent in the underlying OSGi container
static String	**SERVICE_ID** The name of the item containing the service identifier in the CompositeData
static String	**USING_BUNDLES** The name of the item containing the bundles using the service in the CompositeData

Method Summary	
long	**getBundle**(long serviceId) Answer the bundle identifier of the bundle which registered the service
TabularData	**getProperties**(long serviceId) Answer the map of credentials associated with this service
String[]	**getServiceInterfaces**(long serviceId) Answer the list of interfaces that this service implements
TabularData	**getServices**()

Copyright © Oracle Corporation 2009 All Rights Reserved

	Answer the service state of the system in tabular form
long[]	**getUsingBundles**(long serviceId) Answer the list of identifers of the bundles that use the service

Field Detail

BUNDLE_IDENTIFIER

`static final String` **BUNDLE_IDENTIFIER**
The name of the item containing the bundle identifier in the CompositeData

See Also:
Constant Field Values

BUNDLE_LOCATION

`static final String` **BUNDLE_LOCATION**
The name of the item containing the bundle location in the CompositeData

See Also:
Constant Field Values

EVENT_TYPE

`static final String` **EVENT_TYPE**
The name of the item containing the event type in the CompositeData

See Also:
Constant Field Values

OBJECT_CLASS

`static final String` **OBJECT_CLASS**
The name of the item containing the interfaces of the service in the CompositeData

See Also:
Constant Field Values

SERVICE_EVENT_TYPE

Copyright © Oracle Corporation 2009 All Rights Reserved

`static final String` **`SERVICE_EVENT_TYPE`**
The type of the JMX event raised in response to ServiceEvent in the underlying OSGi container

See Also:
Constant Field Values

SERVICE_ID

`static final String` **`SERVICE_ID`**
The name of the item containing the service identifier in the CompositeData

See Also:
Constant Field Values

USING_BUNDLES

`static final String` **`USING_BUNDLES`**
The name of the item containing the bundles using the service in the CompositeData

See Also:
Constant Field Values

SERVICE

`static final String[]` **`SERVICE`**
The item names in the CompositeData representing the service

SERVICE_EVENT

`static final String[]` **`SERVICE_EVENT`**
The item names in the CompositeData representing the ServiceEvent

Method Detail

getServiceInterfaces

```
String[] getServiceInterfaces(long serviceId)
                              throws IOException
```

Answer the list of interfaces that this service implements

Copyright © Oracle Corporation 2009 All Rights Reserved

Parameters:
`serviceId` - - the identifier of the service
Returns:
the list of interfaces
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the service indicated does not exist

getBundle

```
long getBundle(long serviceId)
               throws IOException
```

Answer the bundle identifier of the bundle which registered the service

Parameters:
`serviceId` - - the identifier of the service
Returns:
the identifier for the bundle
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the service indicated does not exist

getProperties

```
TabularData getProperties(long serviceId)
                                                    throws
IOException
```

Answer the map of credentials associated with this service

Parameters:
`serviceId` - - the identifier of the service
Returns:
the table of credentials. These include the standard mandatory service.id and objectClass credentials as defined in the `Constants` interface
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the service indicated does not exist
See Also:
for the details of the TabularType
For each propery entry, the following row is returned
- Property Key - the string key
- Property Value - the stringified version of the property value
- Property Value Type - the type of the property value

Copyright © Oracle Corporation 2009 All Rights Reserved

getServices

`TabularData` **`getServices`**`()`
Answer the service state of the system in tabular form

Returns:
the tabular respresentation of the service state
Throws:
`IOException`
See Also:
for the details of the TabularType
Each row of the returned table represents a single service. For each service, the following row is returned
- identifier - long
- interfaces - String[]
- bundle - long
- using bundles - long[]

See OSGiService for the precise definition of the CompositeType that defines each row of the table.

getUsingBundles

```
long[] getUsingBundles(long serviceId)
                       throws IOException
```
Answer the list of identifers of the bundles that use the service

Parameters:
`serviceId` - - the identifier of the service
Returns:
the list of bundle identifiers
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the service indicated does not exist

Copyright © Oracle Corporation 2009 All Rights Reserved

5.7 Selected Compendium Services

These interfaces provide the remote management agent with control APIs for the OSGi compendium services: Configuration Administration, Permission Administration, Permission Manager, User Manager and the Initial Provisioning Service.

5.7.1 Interface ConfigAdminManagerMBean

org.osgi.jmx.compendium **Interface ConfigAdminManagerMBean**

`public interface` **`ConfigAdminManagerMBean`**

This MBean provides the management interface to the OSGi Configuration Administration Service.

Method Summary	
`void`	**`addProperty`**`(String pid, String name, String value, String type)` Add or update the property for the configuration identified by the supplied pid
`void`	**`addProperty`**`(String pid, String location, String name, String value, String type)` Add or update the property for the configuration identified by the supplied pid and location
`void`	**`addPropertyToConfigurations`**`(String filter, String name, String value, String type)` Add or update the property on all configurations matching the supplied filter
`String`	**`createFactoryConfiguration`**`(String factoryPid)` Create a new configuration instance for the supplied persistent id of the factory, answering the pid of the created configuration
`String`	**`createFactoryConfiguration`**`(String factoryPid, String location)` Creae a factory configuration for the supplied persistent id of the factory and the bundle location bound to bind the created configuration to, answering the pid of the created configuration
`void`	**`delete`**`(String pid)`

Copyright © Oracle Corporation 2009 All Rights Reserved

	Delete the configuration
void	**delete**(String pid, String location) Delete the configuration
void	**deleteConfigurations**(String filter) Delete the configurations matching the filter spec
void	**deleteProperty**(String pid, String key) Delete the property from the configuration
void	**deleteProperty**(String pid, String location, String key) Delete the property from the configuration
void	**deletePropertyFromConfigurations**(String filter, String key) Remove the property from all configurations matching the supplied filter
String	**getBundleLocation**(String pid) Answer the bundle location the configuration is bound to
String	**getFactoryPid**(String pid) Answer the factory pid if the configuration is a factory configuration, null otherwise.
String	**getFactoryPid**(String pid, String location) Answer the factory pid if the configuration is a factory configuration, null otherwise.
TabularData	**getProperties**(String pid) Answer the credentials of the configuration
TabularData	**getProperties**(String pid, String location) Answer the credentials of the configuration
String[][]	**listConfigurations**(String filter) Answer the list of PID/Location pairs of the configurations managed by this service
void	**setBundleLocation**(String pid, String location) Set the bundle location the configuration is bound to
void	**update**(String pid,

Copyright © Oracle Corporation 2009 All Rights Reserved

	`String location, TabularData properties)` Update the configuration with the supplied properties For each propery entry, the following row is supplied
`void`	**`update`**`(String pid, TabularData properties)` Update the configuration with the supplied properties For each propery entry, the following row is supplied

Method Detail

addProperty

```
void addProperty(String pid,
                 String name,
                 String value,
                 String type)
                 throws IOException
```

Add or update the property for the configuration identified by the supplied pid

Parameters:
`pid` - the persistent id of the configuration
`name` - - the property key to add or update
`value` - - the string encoded property value to add or update
`type` - - the type of the property
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the filter is invalid

addProperty

```
void addProperty(String pid,
                 String location,
                 String name,
                 String value,
                 String type)
                 throws IOException
```

Add or update the property for the configuration identified by the supplied pid and location

Parameters:
`pid` - the persistent id of the configuration
`location` - - the bundle location
`name` - - the property key to add or update
`value` - - the string encoded property value to add or update
`type` - - the type of the property

Copyright © Oracle Corporation 2009 All Rights Reserved

Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the filter is invalid

addPropertyToConfigurations

```
void addPropertyToConfigurations(String filter,
                                 String name,
                                 String value,
                                 String type)
                                 throws IOException
```

Add or update the property on all configurations matching the supplied filter

Parameters:
`filter` - the string representation of the `Filter`
`name` - - the property key to add or update
`value` - - the string encoded property value to add or update
`type` - - the type of the property
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the filter is invalid

createFactoryConfiguration

```
String createFactoryConfiguration(String factoryPid)
                                                  throws IOException
```

Create a new configuration instance for the supplied persistent id of the factory, answering the pid of the created configuration

Parameters:
`factoryPid` - - the persistent id of the factory
Returns:
the pid of the created configuation
Throws:
`IOException` - if the operation failed

createFactoryConfiguration

```
String createFactoryConfiguration(String factoryPid,
                                                  String location)
                                                  throws IOException
```

Creae a factory configuration for the supplied persistent id of the factory and the bundle location bound to bind the created configuration to, answering the pid of the created configuration

Parameters:
`factoryPid` - - the persistent id of the factory

Copyright © Oracle Corporation 2009 All Rights Reserved

`location` - - the bundle location
Returns:
the pid of the created configuation
Throws:
`IOException` - if the operation failed

delete

```
void delete(String pid)
            throws IOException
```
Delete the configuration

Parameters:
`pid` - - the persistent identifier of the configuration
Throws:
`IOException` - if the operation fails

delete

```
void delete(String pid,
            String location)
            throws IOException
```
Delete the configuration

Parameters:
`pid` - - the persistent identifier of the configuration
`location` - - the bundle location
Throws:
`IOException` - if the operation fails

deleteConfigurations

```
void deleteConfigurations(String filter)
                          throws IOException
```
Delete the configurations matching the filter spec

Parameters:
`filter` - the string representation of the `Filter`
Throws:
`IOException` - if the operation failed
`IllegalArgumentException` - if the filter is invalid

deleteProperty

```
void deleteProperty(String pid,
                    String key)
```

Copyright © Oracle Corporation 2009 All Rights Reserved

```
                        throws IOException
```
Delete the property from the configuration

Parameters:
`pid` - - the persistent identifier of the configuration
`key` - the property
Throws:
`IOException` - if the operation fails

deleteProperty

```
void deleteProperty(String pid,
                    String location,
                    String key)
                    throws IOException
```
Delete the property from the configuration

Parameters:
`pid` - - the persistent identifier of the configuration
`location` - - the bundle location
`key` - the property
Throws:
`IOException` - if the operation fails

deletePropertyFromConfigurations

```
void deletePropertyFromConfigurations(String filter,
                                      String key)
                                      throws IOException
```
Remove the property from all configurations matching the supplied filter

Parameters:
`filter` - the string representation of the `Filter`
`key` - the property key to be removed
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the filter is invalid

getBundleLocation

```
String getBundleLocation(String pid)
                                    throws IOException
```
Answer the bundle location the configuration is bound to

Parameters:

Copyright © Oracle Corporation 2009 All Rights Reserved

`pid` - - the persistent identifier of the configuration
Returns:
the bundle location
Throws:
`IOException` - if the operation fails

getFactoryPid

```
String getFactoryPid(String pid)
                                    throws IOException
```
Answer the factory pid if the configuration is a factory configuration, null otherwise.

Parameters:
`pid` - - the persistent identifier of the configuration
Returns:
the factory pid
Throws:
`IOException` - if the operation fails

getFactoryPid

```
String getFactoryPid(String pid,
                                    String location)
                                    throws IOException
```
Answer the factory pid if the configuration is a factory configuration, null otherwise.

Parameters:
`pid` - - the persistent identifier of the configuration
`location` - - the bundle location
Returns:
the factory pid
Throws:
`IOException` - if the operation fails

getProperties

```
TabularData getProperties(String pid)
                                                                throws
IOException
```
Answer the credentials of the configuration

Parameters:
`pid` - - the persistent identifier of the configuration
Returns:

Copyright © Oracle Corporation 2009 All Rights Reserved

the table of credentials
Throws:
`IOException` - if the operation fails
See Also:
for the details of the TabularType
For each propery entry, the following row is returned
- Property Key - the string key
- Property Value - the stringified version of the property value
- Property Value Type - the type of the property value

getProperties

```
TabularData getProperties(String pid,
                          String location)
                          throws
IOException
```

Answer the credentials of the configuration

Parameters:
`pid` - - the persistent identifier of the configuration
`location` - - the bundle location
Returns:
the table of credentials
Throws:
`IOException` - if the operation fails
See Also:
for the details of the TabularType
For each propery entry, the following row is returned
- Property Key - the string key
- Property Value - the stringified version of the property value
- Property Value Type - the type of the property value

listConfigurations

```
String[][] listConfigurations(String filter)
                              throws IOException
```

Answer the list of PID/Location pairs of the configurations managed by this service

Parameters:
`filter` - the string representation of the `Filter`
Returns:
the list of configuration PID/Location pairs
Throws:

Copyright © Oracle Corporation 2009 All Rights Reserved

`IOException` - if the operation failed
`IllegalArgumentException` - if the filter is invalid

setBundleLocation

```
void setBundleLocation(String pid,
                       String location)
                       throws IOException
```

Set the bundle location the configuration is bound to

Parameters:
`pid` - - the persistent identifier of the configuration
`location` - - the bundle location
Throws:
`IOException` - if the operation fails

update

```
void update(String pid,
            TabularData properties)
            throws IOException
```

Update the configuration with the supplied properties For each propery entry, the following row is supplied

Parameters:
`pid` - - the persistent identifier of the configuration
`properties` - - the table of properties
Throws:
`IOException` - if the operation fails
See Also:
for the details of the TabularType

- Property Key - the string key
- Property Value - the stringified version of the property value
- Property Value Type - the type of the property value

update

```
void update(String pid,
            String location,
            TabularData properties)
            throws IOException
```

Update the configuration with the supplied properties For each propery entry, the following row is supplied

Copyright © Oracle Corporation 2009 All Rights Reserved

Parameters:
`pid` - - the persistent identifier of the configuration
`location` - - the bundle location
`properties` - - the table of properties
Throws:
`IOException` - if the operation fails
See Also:
`for the details of the TabularType`

- `Property Key - the string key`
- `Property Value - the stringified version of the property value`
- `Property Value Type - the type of the property value`

Copyright © Oracle Corporation 2009 All Rights Reserved

5.7.2 Interface PermissionManagerMBean

org.osgi.jmx.compendium Interface PermissionManagerMBean

`public interface PermissionManagerMBean`

This MBean represents the OSGi Permission Manager Service

Method Summary

`String[]`	**`getLocations`**`()` Answer the bundle locations that have permissions assigned to them
`String[]`	**`getPermissions`**`(String location)` Answer the list of encoded permissions of the bundle specified by the bundle location
`void`	**`setDefaultPermissions`**`(String[] encodedPermissions)` Set the default permissions assiged to bundle locations that have no assigned permissions
`void`	**`setPermissions`**`(String location, String[] encodedPermissions)` Set the permissions on the bundle specified by the bundle location

Method Detail

getLocations

`String[] getLocations()`
`throws IOException`

Answer the bundle locations that have permissions assigned to them

Returns:
the bundle locations
Throws:
`IOException` - if the operation fails

getPermissions

Copyright © Oracle Corporation 2009 All Rights Reserved

```
String[] getPermissions(String location)
                                    throws IOException
```

Answer the list of encoded permissions of the bundle specified by the bundle location

Parameters:
`location` - - location identifying the bundle
Returns:
the array of String encoded permissions
Throws:
`IOException` - if the operation fails

setDefaultPermissions

```
void setDefaultPermissions(String[] encodedPermissions)
                           throws IOException
```

Set the default permissions assiged to bundle locations that have no assigned permissions

Parameters:
`encodedPermissions` - - the string encoded permissions
Throws:
`IOException` - if the operation fails

setPermissions

```
void setPermissions(String location,
                    String[] encodedPermissions)
                    throws IOException
```

Set the permissions on the bundle specified by the bundle location

Parameters:
`location` - - the location of the bundle
`encodedPermissions` - - the string encoded permissions to set
Throws:
`IOException` - if the operation fails

Copyright © Oracle Corporation 2009 All Rights Reserved

5.7.3 Interface ProvisioningMBean

org.osgi.jmx.compendium Interface ProvisioningMBean

```
public interface ProvisioningMBean
```

Author:
Hal Hildebrand Date: Jan 21, 2008 Time: 10:49:26 AM This MBean represents the management interface to the OSGi Initial Provisioning Service

Method Summary

void	**addInformation**(String zipURL) Processes the ZipInputStream contents of the provided zipURL and extracts information to add to the Provisioning Information dictionary, as well as, install/update and start bundles.
void	**addInformation**(TabularData info) Adds the key/value pairs contained in info to the Provisioning Information dictionary.
TabularData	**getInformation**() Returns a table representing the Provisioning Information Dictionary.
void	**setInformation**(TabularData info) Replaces the Provisioning Information dictionary with the entries of the supplied table.

Method Detail

addInformation

```
void addInformation(String zipURL)
                    throws IOException
```

Processes the ZipInputStream contents of the provided zipURL and extracts information to add to the Provisioning Information dictionary, as well as, install/update and start bundles. This method causes the PROVISIONING_UPDATE_COUNT to be incremented.

Parameters:

Copyright © Oracle Corporation 2009 All Rights Reserved

`zipURL` - the String form of the URL that will be resolved into a `ZipInputStream` which will be used to add key/value pairs to the Provisioning Information dictionary and install and start bundles. If a `ZipEntry` does not have an `Extra` field that corresponds to one of the four defined MIME types (`MIME_STRING`, `MIME_BYTE_ARRAY`,`MIME_BUNDLE`, and `MIME_BUNDLE_URL`) in will be silently ignored.
Throws:
`IOException` - if an error occurs while processing the ZipInputStream of the URL. No additions will be made to the Provisioning Information dictionary and no bundles must be started or installed.

addInformation

```
void addInformation(TabularData info)
                    throws IOException
```

Adds the key/value pairs contained in `info` to the Provisioning Information dictionary. This method causes the `PROVISIONING_UPDATE_COUNT` to be incremented.

Parameters:
`info` - the set of Provisioning Information key/value pairs to add to the Provisioning Information dictionary. Any keys are values that are of an invalid type will be silently ignored.
Throws:
`IOException` - if the operation fails
See Also:
for the details of the TabularType
For each entry in the Provisioning Dictionary, the following row is supplied
- Property Key - the string key
- Property Value - the stringified version of the property value
- Property Value Type - the type of the property value

getInformation

```
TabularData getInformation()
                                                    throws
IOException
```

Returns a table representing the Provisioning Information Dictionary.

Returns:
The table representing the manager dictionary.
Throws:
`IOException` - if the operation fails
See Also:

Copyright © Oracle Corporation 2009 All Rights Reserved

for the details of the TabularType
For each entry in the Provisioning Information Dictionary, the following row is supplied
- Property Key - the string key
- Property Value - the stringified version of the property value
- Property Value Type - the type of the property value

setInformation

```
void setInformation(TabularData info)
                    throws IOException
```

Replaces the Provisioning Information dictionary with the entries of the supplied table. This method causes the `PROVISIONING_UPDATE_COUNT` to be incremented.

Parameters:
`info` - the new set of Provisioning Information key/value pairs. Any keys are values that are of an invalid type will be silently ignored.
Throws:
`IOException` - if the operation fails
See Also:
for the details of the TabularType
For each entry in the table, the following row is supplied
- Property Key - the string key
- Property Value - the stringified version of the property value
- Property Value Type - the type of the property value

Copyright © Oracle Corporation 2009 All Rights Reserved

5.7.4 Interface UserManagerMBean

org.osgi.jmx.compendium Interface UserManagerMBean

`public interface UserManagerMBean`

Author:
Hal Hildebrand Date: Dec 2, 2008 Time: 2:41:26 PM This MBean provides the management interface to the OSGi User Manager Service

Field Summary	
static String[]	AUTHORIZATION
static String	ENCODED_CREDENTIALS
static String	ENCODED_ROLE
static String	ENCODED_USER
static String[]	GROUP
static String	GROUP_MEMBERS
static String	GROUP_REQUIRED_MEMBERS
static String[]	ROLE
static String	ROLE_ENCODED_PROPERTIES
static String	ROLE_NAME
static String	ROLE_NAMES
static String	ROLE_TYPE
static String[]	USER
static String	USER_NAME

Method Summary	
void	addCredential(String key, byte[] value,

Copyright © Oracle Corporation 2009 All Rights Reserved

	String username) Add credentials to a user, associated with the supplied key
void	addCredential(String key, String value, String username) Add credentials to a user, associated with the supplied key
boolean	addMember(String groupname, String rolename) Add a role to the group
void	addProperty(String key, byte[] value, String rolename) Add or update a property on a role
void	addProperty(String key, String value, String rolename) Add or update a property on a role
boolean	addRequiredMember(String groupname, String rolename) Add a required member to the group
void	createGroup(String name) Create a Group
void	createUser(String name) Create a User
CompositeData	getAuthorization(String user) Answer the authorization for the user name
TabularData	getCredentials(String username) Answer the credentials associated with a user
CompositeData	getGroup(String groupname) Answer the Group associated with the groupname
String[]	getGroups() Answer the list of group names
String[]	getGroups(String filter) Answer the list of group names
String[]	getImpliedRoles(String username) Answer the list of implied roles for a user
String[]	getMembers(String groupname) Answer the the user names which are members of the group
TabularData	getProperties(String rolename)

Copyright © Oracle Corporation 2009 All Rights Reserved

	Answer the credentials associated with a role
String[]	**getRequiredMembers**(String groupname) Answer the list of user names which are required members of this group
CompositeData	**getRole**(String name) Answer the role associated with a name
String[]	**getRoles**() Answer the list of role names in the User Admin database
String[]	**getRoles**(String filter) Answer the list of role names which match the supplied filter
CompositeData	**getUser**(String username) Answer the User associated with the username
String	**getUser**(String key, String value) Answer the user name with the given property key-value pair from the User Admin service database.
String[]	**getUsers**() Answer the list of user names in the User Admin database
String[]	**getUsers**(String filter) Answer the list of user names in the User Admin database
void	**removeCredential**(String key, String username) Remove the credentials associated with the key for the user
boolean	**removeMember**(String groupname, String rolename) Remove a role from the group
void	**removeProperty**(String key, String rolename) Remove a property from a role
boolean	**removeRole**(String name) Remove the Role associated with the name

Field Detail

ROLE_NAME

```
static final String ROLE_NAME
```

See Also:

Copyright © Oracle Corporation 2009 All Rights Reserved

Constant Field Values

ROLE_TYPE

`static final String` **`ROLE_TYPE`**
See Also:
Constant Field Values

ROLE_ENCODED_PROPERTIES

`static final String` **`ROLE_ENCODED_PROPERTIES`**
See Also:
Constant Field Values

ENCODED_USER

`static final String` **`ENCODED_USER`**
See Also:
Constant Field Values

GROUP_MEMBERS

`static final String` **`GROUP_MEMBERS`**
See Also:
Constant Field Values

GROUP_REQUIRED_MEMBERS

`static final String` **`GROUP_REQUIRED_MEMBERS`**
See Also:
Constant Field Values

USER_NAME

`static final String` **`USER_NAME`**
See Also:
Constant Field Values

ROLE_NAMES

`static final String` **`ROLE_NAMES`**
See Also:
Constant Field Values

ENCODED_ROLE

Copyright © Oracle Corporation 2009 All Rights Reserved

```
static final String ENCODED_ROLE
```
See Also:
Constant Field Values

ENCODED_CREDENTIALS

```
static final String ENCODED_CREDENTIALS
```
See Also:
Constant Field Values

AUTHORIZATION

```
static final String[] AUTHORIZATION
```

USER

```
static final String[] USER
```

ROLE

```
static final String[] ROLE
```

GROUP

```
static final String[] GROUP
```

Method Detail

addCredential

```
void addCredential(String key,
                   byte[] value,
                   String username)
                   throws IOException
```
Add credentials to a user, associated with the supplied key

Parameters:
`key` -
`value` -
`username` -
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the username is not a User

Copyright © Oracle Corporation 2009 All Rights Reserved

addCredential

```
void addCredential(String key,
                   String value,
                   String username)
                   throws IOException
```

Add credentials to a user, associated with the supplied key

Parameters:
`key` -
`value` -
`username` -
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the username is not a User

addMember

```
boolean addMember(String groupname,
                  String rolename)
                  throws IOException
```

Add a role to the group

Parameters:
`groupname` -
`rolename` -
Returns:
true if the role was added to the group
Throws:
`IOException` - if the operation fails

addProperty

```
void addProperty(String key,
                 String value,
                 String rolename)
                 throws IOException
```

Add or update a property on a role

Parameters:
`key` - - the property key
`value` - - the String property value
`rolename` - - the role name
Throws:
`IOException` - if the operation fails

Copyright © Oracle Corporation 2009 All Rights Reserved

addProperty

```
void addProperty(String key,
                 byte[] value,
                 String rolename)
                 throws IOException
```

Add or update a property on a role

Parameters:
`key` - - the property key
`value` - - the byte[] property value
`rolename` - - the role name
Throws:
`IOException` - if the operation fails

addRequiredMember

```
boolean addRequiredMember(String groupname,
                          String rolename)
                          throws IOException
```

Add a required member to the group

Parameters:
`groupname` -
`rolename` -
Returns:
true if the role was added to the group
Throws:
`IOException` - if the operation fails

createUser

```
void createUser(String name)
                throws IOException
```

Create a User

Parameters:
`name` - - the user to create
Throws:
`IOException` - if the operation fails

createGroup

```
void createGroup(String name)
                 throws IOException
```

Create a Group

Copyright © Oracle Corporation 2009 All Rights Reserved

Parameters:
`name` - - the group to create
Throws:
`IOException` - if the operation fails

getAuthorization

`CompositeData` **`getAuthorization`**`(String user)`
`throws`
`IOException`
Answer the authorization for the user name

Parameters:
`user` -
Returns:
the Authorization
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the username is not a User
See Also:
`for the details of the CompositeType`

getCredentials

`TabularData` **`getCredentials`**`(String username)`
`throws`
`IOException`
Answer the credentials associated with a user

Parameters:
`username` -
Returns:
the credentials associated with the user
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the username is not a User
See Also:
`for the details of the TabularType`

Copyright © Oracle Corporation 2009 All Rights Reserved

getGroup

```
CompositeData getGroup(String groupname)
                                                    throws IOException
```

Answer the Group associated with the groupname

Parameters:
`groupname` -
Returns:
the Group
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the groupname is not a Group
See Also:
for the details of the CompositeType

getGroups

```
String[] getGroups()
                                    throws IOException
```

Answer the list of group names

Returns:
the list of group names
Throws:
`IOException` - if the operation fails

getGroups

```
String[] getGroups(String filter)
                                    throws IOException
```

Answer the list of group names

Parameters:
`filter` - - the filter to apply
Returns:
the list of group names
Throws:
`IOException` - if the operation fails

getImpliedRoles

Copyright © Oracle Corporation 2009 All Rights Reserved

```
String[] getImpliedRoles(String username)
                                    throws IOException
```

Answer the list of implied roles for a user

Parameters:
`username` -
Returns:
the list of role names
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the username is not a User

getMembers

```
String[] getMembers(String groupname)
                                  throws IOException
```

Answer the the user names which are members of the group

Parameters:
`groupname` -
Returns:
the list of user names
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the groupname is not a group

getProperties

```
TabularData getProperties(String rolename)
                                                    throws
IOException
```

Answer the credentials associated with a role

Parameters:
`rolename` -
Returns:
the credentials associated with the role
Throws:
`IOException` - if the operation fails
See Also:
`for the details of the TabularType`

getRequiredMembers

Copyright © Oracle Corporation 2009 All Rights Reserved

```
String[] getRequiredMembers(String groupname)
                                                    throws IOException
```

Answer the list of user names which are required members of this group

Parameters:
`groupname` -
Returns:
the list of user names
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the groupname is not a group

getRole

```
CompositeData getRole(String name)
                                                              throws IOException
```

Answer the role associated with a name

Parameters:
`name` -
Returns:
the Role
Throws:
`IOException` - if the operation fails
See Also:
for the details of the CompositeType

getRoles

```
String[] getRoles()
                                  throws IOException
```

Answer the list of role names in the User Admin database

Returns:
the list of role names
Throws:
`IOException` - if the operation fails

getRoles

```
String[] getRoles(String filter)
                                  throws IOException
```

Copyright © Oracle Corporation 2009 All Rights Reserved

Answer the list of role names which match the supplied filter

Parameters:
`filter` - - the string representation of the `Filter`
Returns:
the list the role names
Throws:
`IOException` - if the operation fails

getUser

```
CompositeData getUser(String username)
                                                    throws IOException
```

Answer the User associated with the username

Parameters:
`username` -
Returns:
the User
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the username is not a User
See Also:
for the details of the CompositeType

getUser

```
String getUser(String key,
                           String value)
                           throws IOException
```

Answer the user name with the given property key-value pair from the User Admin service database.

Parameters:
`key` - - the key to compare
`value` - - the value to compare
Returns:
the User
Throws:
`IOException` - if the operation fails

getUsers

Copyright © Oracle Corporation 2009 All Rights Reserved

```
String[] getUsers()
                                throws IOException
```
Answer the list of user names in the User Admin database

Returns:
the list of user names
Throws:
`IOException` - if the operation fails

getUsers

```
String[] getUsers(String filter)
                                throws IOException
```
Answer the list of user names in the User Admin database

Parameters:
`filter` - - the filter to apply
Returns:
the list of user names
Throws:
`IOException` - if the operation fails

removeCredential

```
void removeCredential(String key,
                      String username)
                      throws IOException
```
Remove the credentials associated with the key for the user

Parameters:
`key` -
`username` -
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the username is not a User

removeMember

```
boolean removeMember(String groupname,
                     String rolename)
                     throws IOException
```
Remove a role from the group

Parameters:
`groupname` -

Copyright © Oracle Corporation 2009 All Rights Reserved

`rolename` -
Returns:
true if the role was removed from the group
Throws:
`IOException` - if the operation fails
`IllegalArgumentException` - if the groupname is not a Group

removeProperty

```
void removeProperty(String key,
                    String rolename)
                    throws IOException
```

Remove a property from a role

Parameters:
`key` -
`rolename` -
Throws:
`IOException` - if the operation fails

removeRole

```
boolean removeRole(String name)
                   throws IOException
```

Remove the Role associated with the name

Parameters:
`name` -
Returns:
true if the remove succeeded
Throws:
`IOException` - if the operation fails

6 Considered Alternatives

This section explores various mechanisms for exposing the OSGi framework API into JMX and documents the potential shortcomings of each.

6.1 Direct Translation of the OSGi Framework APIs

A straightforward approach is to simply replicate the OSGi framework APIs, directly translating concrete classes into interfaces and augmenting existing interfaces to transform them into a JMX compliant system. This approach provides a very RMI like interface to underlying framework APIs, in that the exposed objects have a direct one to

Copyright © Oracle Corporation 2009 All Rights Reserved

one relationship with the framework artifacts which replicates the underlying APIs as closely as possible. The problem with this approach is that the underlying OSGi framework APIs are designed for in process, direct manipulation and are not abstracted and designed to facilitate the remote management of the framework. Operations which are completely natural when performed in the same process become cumbersome and impractical when viewed from the perspective of a remote management agent.

6.2 Automatic JMX Translation of OSGi Framework and Services

There are a number of very nice systems which provide various degrees of transparently publishing existing services and frameworks into JMX. These system have the advantage of not requiring any changes off the underlying systems being exposed through the JMX framework, however they share the same disadvantages of a direct translation, discussed in the previous section, in that the systems were not designed with remote management in mind and nothing these automatic translation systems can do will change that.

6.3 JMX Translation of Management Technology Neutral Refactoring

The idea here is that there would be value in coming up with a technology neutral facade which could then be consumed by JMX. There is potentially a lot of value in providing a technology “neutral” management API in that this could be reused in other technologies as well as JMX. However, the artifacts of a JMX compliant interface are largely Java interfaces and perhaps some concrete classes for interchange. This means that implementations can make use of techniques such as the *javax.management.StandardMBean*[5] which allows straight forward implementation of the interfaces with JMX standard MBean techniques without polluting the package namespace with the JMX implementation. Consequently, the interfaces defined for JMX management of OSGi do not have any JMX bleed through which would prevent it from being in other management frameworks.

7 Security Considerations

The management interfaces in this specification are designed for use with the JMX framework. JMX has its own security permission framework as well as specification of remote authentication and authorization. Consequently, all security considerations are delegated to the enclosing JMX framework which hosts these interfaces

8 Document Support

8.1 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

[3]. Java Management Extensions (JMX) Technology Overview
http://java.sun.com/j2se/1.5.0/docs/guide/jmx/overview/JMXoverviewTOC.html

Copyright © Oracle Corporation 2009 All Rights Reserved

[4]. JavaTM Management Extensions (JMXTM)API Specification
http://java.sun.com/j2se/1.5.0/docs/guide/jmx/spec.html

[5]. Javax.management.StandardMBean
http://java.sun.com/javase/6/docs/api/javax/management/StandardMBean.html

[6]. Using JConsole to Monitor Applications
http://java.sun.com/developer/technicalArticles/J2SE/jconsole.html

8.2 Author's Address

Name	Hal Hildebrand
Company	Oracle
Address	500 Oracle Parkway, M/S 2op946, Redwood City, CA 94065
Voice	+1 650 563 9646
e-mail	hal.hildebrand@oracle.com

8.3 Acronyms and Abbreviations

8.4 End of Document

Copyright © Oracle Corporation 2009 All Rights Reserved