RFC 185 Data Transfer Objects

Final

52 Pages

Abstract

Define a set of data-only objects to represent runtime objects. The types will have limited behavior to support easy serialization and use by management agents to communicate with external systems.

0 Document Information

0.1 License

0.2 Trademarks

0.3 Feedback

This document can be downloaded from the OSGi Alliance design repository at https://github.com/osgi/design The public can provide feedback about this document by opening a bug at https://www.osgi.org/bugzilla/.

0.4 Table of Contents

0.5 Terminology and Document Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in 1.

`Source code is shown in this typeface.`

0.6 Revision History

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

Revision	Date	Comments
Initial	9 Jun 2012	Initial draft.
2nd draft	13 Jun 2012	Updated after CPEG f2f in NYC
3rd draft	11 Sep 2012	Added new DTOs to complete the DTOs for the Core framework.
4th draft	24 Oct 2012	Updated based upon comments from Basel F2F.
5th draft	29 Oct 2012	Updated based upon comments from 25 Oct 2012 CPEG call.
6th draft	5 Nov 2012	David Bosschaert (Red Hat) adding JMX section.
7th draft	9 Nov 2012	Updates from 6 Nov Orlando F2F.
8th draft	5 Dec 2012	Updates for bugs 2474 (toString) and 2475 (List, Set and Map). DTO is now an abstract class with toString behavior. Serializable is also removed since it cannot be supported in the base class alone. See bug 2475.

Revision	Date	Comments
9th draft	12 Dec 2012	While implementing, it became clear that FrameworkDTO needed to be adapted from the system bundle (instead of any bundle). This because a bundle context is needed to get services. Random bundles may not have valid contexts. Also there is the issue of hooks filtering services and bundles. Limiting this to the system bundles simplifies things and ensure the same DTO view rather than an empty or filtered one. Javadoc also updated to indicate that some adaptations don't apply after the bundle is uninstalled.
10th draft	3 Feb 2014	Replace BundleRevisionsDTO with adapting to BundleRevisionDTO[] and BundleWiringsDTO with adapting to BundleWiringDTO[]. Restructure wiring DTOs to remove cycles. The set of referenced DTOs is pushed up to the top level BundleWiringDTO and each DTO is assigned a transient identifier that can be used as a reference by other DTOs. An example transient identifier could be the identity hash code of the underlying runtime object. Since CapabilityDTO and RequirementDTO object can be large, due to the contained directives and attributes maps, we change to use ref DTOs in the wiring DTO types. The ref DTOs refer to the CapabilityDTO/RequirementDTO objects in the BundleRevisionDTO.
Final	7 Feb 2014	The naming convention for DTO packages has been changed to place the dto segment at the end of the package name rather than in the middle.

1 Introduction

The OSGi API is rich and introspective. Since the API has a lot of behavior and is not designed for serialization, each management model must design its own representation of the relevant OSGi objects for transport to the remote management system. We see this in JMX, DMT and also in REST. Having standard, simple, easy to serialize and deserialize objects which represent the relevant OSGi object will make it easier for the management model to keep up with changes in the OSGi API.

2 Application Domain

While OSGi has a rich API for local management of bundles, services, etc., each management model must define how this OSGi objects are represented for communication with remote management systems. JMX must define the Mbeans, DMT must define the tree representation, REST must define the request/response payload.

The OSGi API continues to evolve and at each update of the OSGi API, the management systems will all need to update their representation of the OSGi objects.

3 Problem Description

Since each management model defines its own representation of the OSGi objects, each management model specification will need to be updated whenever some new feature is added to the OSGi API. A common, shared representation will reduce the effort needed by each management model specification to track changes the OSGi API.

4 Requirements

DTO-0001 – DTOs must be easily serializable. That is, no special serialization/deserialization logic must be required. Serialization must be possible simply by introspecting the DTO objects and object graphs must be a tree.

DTO-0002 – DTOs must have no behavior. That is, no methods other than the default public constructor.

DTO-0003 – DTOs must have only public fields.

DTO-0004 – The types of the fields in a DTO must be one of:

- primitive numerical types or their wrapper classes (e.g. int, Long)

- boolean or Boolean
- String
- a DTO
- Arrays
- Lists
- Sets
- Maps

No other types are permitted. The aggregates (arrays, Lists, Sets and Maps) may only contain any of the allowed types including aggregates.

DTO-0005 – A DTO may extend another DTO.

DTO-0006 – A mechanism must be provided to create DTO objects for the real objects they represent.

5 Technical Solution

There are two main parts to Data Transfer Objects: the design of the data structures and how to obtain instances of the data structures from the framework or other OSGi service.

5.1 Data Transfer Object Design

A Data Transfer Object [3]. is used to capture the state of a related object in a form suitable for easy transfer to some receiver. The receiver can be in the same JVM but is more likely in another process or on another system that is remote.

All DTOs are easily serializable having only public fields of primitive types and their wrapper classes, Strings, and DTOs. List, Set, Map and array aggregates may also be used. The aggregates must only hold objects of the listed types or aggregates. All DTOs must extend the `org.osgi.dto.DTO` abstract base class. DTOs are public classes with no methods (other than the compiler supplied default constructor) having only public fields limited to the easily serializable types mentioned above.

The `org.osgi.dto` package defines the basic rules and base DTO type which is extended by other DTOs.

5.1.1 DTO Naming Conventions

DTOs should follow a naming convention for the package containing the DTO as well as the DTO type.

For the package name, DTOs should be in a package that ends with `.dto`. So for a DTO for a type in the `org.osgi.service.foo` package, the proper DTO package name is `org.osgi.service.foo.dto`.

The name of the DTO type should be the name of the type for which the DTO provides a snapshot of the state followed by "`DTO`". So for a type `Widget`, the DTO for that type should be `WidgetDTO`. Sometime the entity for which the DTO provides state is not represented by a type; for example, Framework. In this case, the name of entity with a DTO suffix should be used: `FrameworkDTO`.

Putting both the package and type DTO naming conventions together: The DTO for `org.osgi.service.foo.Widget` would be `org.osgi.service.foo.dto.WidgetDTO`.

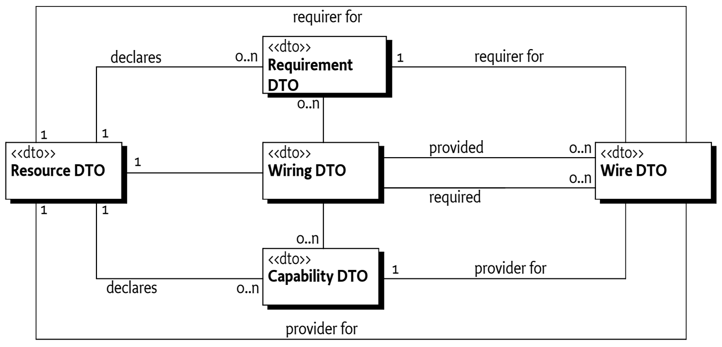
5.1.2 Core DTOs

DTOs are defined for the key framework objects: Bundle, "framework", ServiceReference, resource types, startlevel types and wiring types.

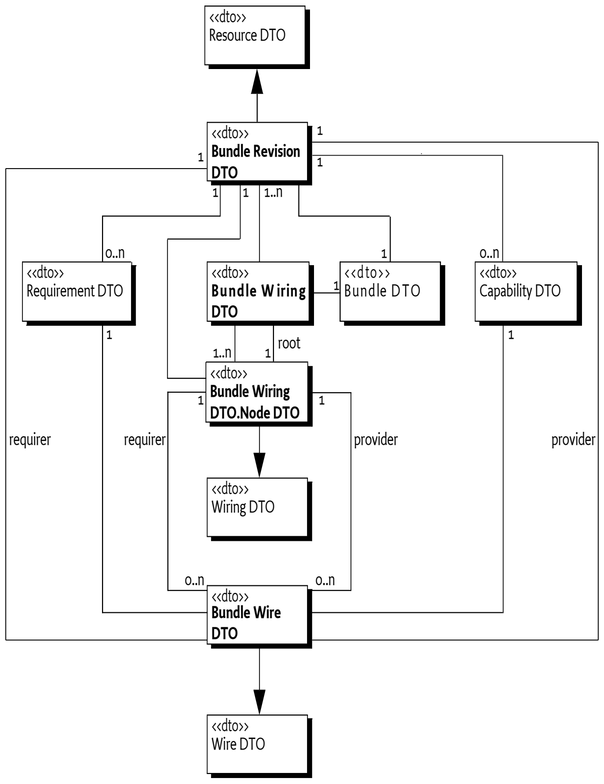
BundleDTO provides information about a single bundle. FrameworkDTO provides the list of installed bundles, the registered services and the launch properties of a single framework. ServiceReferenceDTO provides, for a single service, the service properties, the bundle which registered the service and the bundles using the service.

BundleStartLevelDTO provides the start level information about a single bundle. FrameworkStartLevelDTO provides the start level information about a single framework.

CapabilityDTO, RequirementDTO, ResourceDTO, WiringDTO and WireDTO provide the resource package equivalent view of capabilities and requirements wiring information. The following figure shows the effective relationship between the DTO types. Since the graph of Data Transfer Objects must be a tree, some references are indirect.



BundleRevisionDTO, BundleWiringDTO, BundleWireDTO provide the wiring package equivalent view of capabilities and requirements wiring information. The following figure shows the effective relationship between the DTO types. Since the graph of Data Transfer Objects must be a tree, some references are indirect.



5.2 Obtaining Data Transfer Objects

5.2.1 Core DTOs

The framework must supply DTO objects via Bundle.adapt.

An installed Bundle object can be adapted to: BundleDTO, ServiceReferenceDTO[], BundleStartLevelDTO, BundleRevisionDTO, BundleRevisionDTO[], BundleWiringDTO, and BundleWiringDTO[]. The System Bundle object can be adapted to: FrameworkDTO and FrameworkStartLevelDTO.

A FrameworkDTO object can be used to obtain all the BundleDTOs and ServiceReferenceDTOs for installed bundles and registered services.

For example:

```
// DTO for the bundle
BundleDTO bundleDTO = bundle.adapt(BundleDTO.class);
```

```
// DTO for the current bundle wiring
BundleWiringDTO bundleWiringDTO = bundle.adapt(BundleWiringDTO.class);

// DTO for the current bundle revision
BundleRevisionDTO bundleRevisionDTO = bundle.adapt(BundleRevisionDTO.class);
```

5.3 Examples of DTO usage

5.3.1 REST

RFC 182 defines a REST interface to the OSGi framework. The DTO objects defined in this RFC can be used to create the representations for the REST interfaces.

A REST request to get the bundle information for bundle 1 (GET framework/bundle/1) can obtain the representation information using the BundleDTO.

```
long id = getBundleIdFromURI(requestURI);
BundleDTO bundleDTO = getContext().getBundle(id).adapt(BundleDTO.class);
String response = jsonSerializer(bundleDTO); // serialize to JSON (or XML)
```

5.3.2 JMX

The JMX spec defines a JMX interface to the OSGi framework. The DTO objects defined in this RFC can be used to obtain state information to be used by the JMX MBeans.

To expose the Framework DTOs a new JMX MBean needs to be defined that provides access to these DTOs. For example:

```
public interface FrameworkMBean {
    CompositeData[] getBundles();
    CompositeData getBundle(long id);
    CompositeData[] getServices();
    // … etc …
}
```

Instead of the plain DTO object, JMX-OpenBean versions of the objects are provided through this API. This means that Open Type supported simple types (as defined in javax.management.openmbean.OpenType.-ALLOWED_CLASSNAMES_LIST) can be used as-is, but embedded DTOs and maps need to be transformed into JMX structures, as listed in the following table:

DTO data type	***JMX data type***
simple type (as supported by JMX Open Types)	javax.management.openmbean.SimpleType constant
Map	TabularType
custom DTO	CompositeType
custom DTO []	CompositeType[]

Given a certain DTO, a fairly straightforward generic transformation can be defined to produce JMX friendly data structures, this can be achieved by introspecting the DTOs using Java reflection and generating CompositeType definitions and CompositeData objects from them. Composite types and Tabulary types support nesting so nested DTOs can be supported.

Non-framework DTOs

For DTOs that provide information regarding an Enterprise, Residential or other Compendium specification, root MBeans will still be necessary in the MBean registry, however these MBeans can simply provide access to JMX views over the relevant DTOs which can be automatically produced from the DTO definition.

Modifying the Framework state

The DTOs don't provide a mechanism to change the state of the framework (or any other component) so in order to support this, specific APIs still need to be provided by the JMX MBeans.

Maintenance

Using DTOs will significantly reduce the maintenance required to provide viewing capabilities into the framework and into other components that expose themselves as DTOs, as the DTO definitions can be used to generate JMX OpenBeans suitable for a JMX management agent.

Maintenance is still needed for APIs that alter the framework state.

5.3.3 Residential DMT

The Residential DMT spec defines a Device Management Tree (DMT) interface to the OSGi framework. The DTO objects defined in this RFC can be used to obtain state information to be used to populate information in the Residential DMT.

```
// $/Framework/StartLevel node value

FrameworkStartLevelDTO fslDTO =
getContext().getBundle(0).adapt(FrameworkStartLevelDTO.class);
return new DmtData(FrameworkStartLevelDTO.startLevel);
```

6 Javadoc

OSGi Javadoc

2/7/14 12:57 PM

Package Summary		Page
org.osgi.dto	OSGi Data Transfer Object Package Version 1.0.	*13*
org.osgi.framework.dto	OSGi Data Transfer Object Framework Package Version 1.8.	*15*
org.osgi.framework.startlevel.dto	OSGi Data Transfer Object Framework Start Level Package Version 1.0.	*22*
org.osgi.framework.wiring.dto	OSGi Data Transfer Object Framework Wiring Package Version 1.2.	*27*
org.osgi.resource.dto	OSGi Data Transfer Object Resource Package Version 1.0.	*36*

Package org.osgi.dto

```
@org.osgi.annotation.versioning.Version(value="1.0")
```

OSGi Data Transfer Object Package Version 1.0.

See:

Description

Class Summary		*Page*
DTO	Super type for Data Transfer Objects.	*14*

Package org.osgi.dto Description

OSGi Data Transfer Object Package Version 1.0.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. This package has two types of users: the consumers that use the API in this package and the providers that implement the API in this package.

Example import for consumers using the API in this package:

```
Import-Package: org.osgi.dto; version="[1.0,2.0)"
```

Example import for providers implementing the API in this package:

```
Import-Package: org.osgi.dto; version="[1.0,1.1)"
```

Class DTO

org.osgi.dto

```
java.lang.Object
  └org.osgi.dto.DTO
```

Direct Known Subclasses:

BundleDTO, BundleStartLevelDTO, BundleWiringDTO, CapabilityDTO, CapabilityRefDTO, FrameworkDTO, FrameworkStartLevelDTO, RequirementDTO, RequirementRefDTO, ResourceDTO, ServiceReferenceDTO, WireDTO, WiringDTO

```
abstract public class DTO
extends Object
```

Super type for Data Transfer Objects. All data transfer objects are easily serializable having only public fields of primitive types and their wrapper classes, Strings, and DTOs. List, Set, Map and array aggregates may also be used. The aggregates must only hold objects of the listed types or aggregates.

NotThreadSafe

Constructor Summary	*Page*
`DTO()`	*14*

Method Summary		*Page*
`String`	`toString()` Return a string representation of this DTO suitable for use when debugging.	*14*

Constructor Detail

DTO

```
public DTO()
```

Method Detail

toString

```
public String toString()
```

Return a string representation of this DTO suitable for use when debugging.

The format of the string representation is not specified and subject to change.

Overrides:
`toString` in class `Object`

Returns:
A string representation of this DTO suitable for use when debugging.

Package org.osgi.framework.dto

```
@org.osgi.annotation.versioning.Version(value="1.8")
```

OSGi Data Transfer Object Framework Package Version 1.8.

See:

[Description]

Class Summary		Page
BundleDTO	Data Transfer Object for a Bundle.	*16*
FrameworkDTO	Data Transfer Object for a Framework.	*18*
ServiceReferenceDTO	Data Transfer Object for a ServiceReference.	*20*

Package org.osgi.framework.dto Description

OSGi Data Transfer Object Framework Package Version 1.8.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. This package has two types of users: the consumers that use the API in this package and the providers that implement the API in this package.

Example import for consumers using the API in this package:

```
Import-Package: org.osgi.framework.dto; version="[1.8,2.0)"
```

Example import for providers implementing the API in this package:

```
Import-Package: org.osgi.framework.dto; version="[1.8,1.9)"
```

Class BundleDTO

org.osgi.framework.dto

```
java.lang.Object
  └org.osgi.dto.DTO
      └org.osgi.framework.dto.BundleDTO
```

```
public class BundleDTO
extends DTO
```

Data Transfer Object for a Bundle.

A Bundle can be adapted to provide a `BundleDTO` for the Bundle.

NotThreadSafe

Field Summary		Page
long	**id** The bundle's unique identifier.	16
long	**lastModified** The time when the bundle was last modified.	16
int	**state** The bundle's state.	17
String	**symbolicName** The bundle's symbolic name.	17
String	**version** The bundle's version.	17

Constructor Summary	Page
BundleDTO()	17

Methods inherited from class org.osgi.dto.DTO
toString

Field Detail

id

```
public long id
```

The bundle's unique identifier.

See Also:
`org.osgi.framework.Bundle.getBundleId()`

lastModified

```
public long lastModified
```

The time when the bundle was last modified.

See Also:
`org.osgi.framework.Bundle.getLastModified()`

---

state

`public int **state**`

The bundle's state.

See Also:
`org.osgi.framework.Bundle.getState()`

---

symbolicName

`public String **symbolicName**`

The bundle's symbolic name.

See Also:
`org.osgi.framework.Bundle.getSymbolicName()`

---

version

`public String **version**`

The bundle's version.

See Also:
`org.osgi.framework.Bundle.getVersion()`

Constructor Detail

BundleDTO

`public **BundleDTO**()`

Class FrameworkDTO

org.osgi.framework.dto

```
java.lang.Object
  └org.osgi.dto.DTO
      └org.osgi.framework.dto.FrameworkDTO
```

```
public class FrameworkDTO
extends DTO
```

Data Transfer Object for a Framework.

The System Bundle can be adapted to provide a `FrameworkDTO` for the framework of the system bundle. A `FrameworkDTO` obtained from a framework will contain only the launch properties of the framework. These properties will not include the System properties.

NotThreadSafe

Field Summary		Page
List<BundleDTO>	**bundles** The bundles that are installed in the framework.	18
Map<String,Object>	**properties** The launch properties of the framework.	18
List<ServiceReferenceDTO>	**services** The services that are registered in the framework.	19

Constructor Summary	Page
FrameworkDTO()	19

Methods inherited from class org.osgi.dto.DTO
toString

Field Detail

bundles

`public List<BundleDTO> bundles`

The bundles that are installed in the framework.

See Also:
`org.osgi.framework.BundleContext.getBundles()`

properties

`public Map<String,Object> properties`

The launch properties of the framework. The value type must be a numerical type, Boolean, String, DTO or an array of any of the former.

See Also:

`org.osgi.framework.BundleContext.getProperty(String)`

---

services

`public List<ServiceReferenceDTO> services`

The services that are registered in the framework.

See Also:

`org.osgi.framework.BundleContext.getServiceReferences(String, String)`

Constructor Detail

FrameworkDTO

`public FrameworkDTO()`

Class ServiceReferenceDTO

org.osgi.framework.dto

```
java.lang.Object
  └ org.osgi.dto.DTO
      └ org.osgi.framework.dto.ServiceReferenceDTO
```

```
public class ServiceReferenceDTO
extends DTO
```

Data Transfer Object for a ServiceReference.

`ServiceReferenceDTO`s for all registered services can be obtained from a `FrameworkDTO`. An installed Bundle can be adapted to provide a `ServiceReferenceDTO[]` of the services registered by the Bundle. A `ServiceReferenceDTO` obtained from a framework must convert service property values which are not valid value types for DTOs to type `String` using `String.valueOf(Object)`.

NotThreadSafe

Field Summary		*Page*
`long`	**`bundle`** The id of the bundle that registered the service.	*20*
`long`	**`id`** The `service.id` of the service.	*20*
`Map<String,Object>`	**`properties`** The properties for the service.	*21*
`long[]`	**`usingBundles`** The ids of the bundles that are using the service.	*21*

Constructor Summary	*Page*
`ServiceReferenceDTO``()`	*21*

Methods inherited from class org.osgi.dto.DTO
`toString`

Field Detail

id

```
public long id
```

The `service.id` of the service.

See Also:
`org.osgi.framework.Constants.SERVICE_ID`

bundle

```
public long bundle
```

The id of the bundle that registered the service.

See Also:

`org.osgi.framework.ServiceReference.getBundle()`

---

properties

`public Map<String,Object> properties`

The properties for the service. The value type must be a numerical type, Boolean, String, DTO or an array of any of the former.

See Also:

`org.osgi.framework.ServiceReference.getProperty(String)`

---

usingBundles

`public long[] usingBundles`

The ids of the bundles that are using the service.

See Also:

`org.osgi.framework.ServiceReference.getUsingBundles()`

Constructor Detail

ServiceReferenceDTO

`public ServiceReferenceDTO()`

Package org.osgi.framework.startlevel.dto

```
@org.osgi.annotation.versioning.Version(value="1.0")
```

OSGi Data Transfer Object Framework Start Level Package Version 1.0.

See:
Description

Class Summary		*Page*
BundleStartLevelDTO	Data Transfer Object for a BundleStartLevel.	*23*
FrameworkStartLevelDTO	Data Transfer Object for a FrameworkStartLevel.	*25*

Package org.osgi.framework.startlevel.dto Description

OSGi Data Transfer Object Framework Start Level Package Version 1.0.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. This package has two types of users: the consumers that use the API in this package and the providers that implement the API in this package.

Example import for consumers using the API in this package:

```
Import-Package: org.osgi.framework.startlevel.dto; version="[1.0,2.0)"
```

Example import for providers implementing the API in this package:

```
Import-Package: org.osgi.framework.startlevel.dto; version="[1.0,1.1)"
```

Class BundleStartLevelDTO

org.osgi.framework.startlevel.dto

```
java.lang.Object
  └org.osgi.dto.DTO
      └org.osgi.framework.startlevel.dto.BundleStartLevelDTO
```

```
public class BundleStartLevelDTO
extends DTO
```

Data Transfer Object for a BundleStartLevel.

An installed Bundle can be adapted to provide a `BundleStartLevelDTO` for the Bundle.

NotThreadSafe

Field Summary		Page
`boolean`	**`activationPolicyUsed`** The bundle's autostart setting indicates that the activation policy declared in the bundle manifest must be used.	*24*
`long`	**`bundle`** The id of the bundle associated with this start level.	*23*
`boolean`	**`persistentlyStarted`** The bundle's autostart setting indicates it must be started.	*24*
`int`	**`startLevel`** The assigned start level value for the bundle.	*23*

Constructor Summary	Page
`BundleStartLevelDTO``()`	*24*

Methods inherited from class org.osgi.dto.DTO
`toString`

Field Detail

bundle

```
public long bundle
```

The id of the bundle associated with this start level.

See Also:
`org.osgi.framework.BundleReference.getBundle()`

startLevel

```
public int startLevel
```

The assigned start level value for the bundle.

See Also:

`org.osgi.framework.startlevel.BundleStartLevel.getStartLevel()`

---

activationPolicyUsed

```
public boolean activationPolicyUsed
```

The bundle's autostart setting indicates that the activation policy declared in the bundle manifest must be used.

See Also:

`org.osgi.framework.startlevel.BundleStartLevel.isActivationPolicyUsed()`

---

persistentlyStarted

```
public boolean persistentlyStarted
```

The bundle's autostart setting indicates it must be started.

See Also:

`org.osgi.framework.startlevel.BundleStartLevel.isPersistentlyStarted()`

Constructor Detail

BundleStartLevelDTO

```
public BundleStartLevelDTO()
```

Class FrameworkStartLevelDTO

org.osgi.framework.startlevel.dto

```
java.lang.Object
  └org.osgi.dto.DTO
      └org.osgi.framework.startlevel.dto.FrameworkStartLevelDTO
```

```
public class FrameworkStartLevelDTO
extends DTO
```

Data Transfer Object for a FrameworkStartLevel.

The System Bundle can be adapted to provide a `FrameworkStartLevelDTO` for the framework of the Bundle.

NotThreadSafe

Field Summary		Page
`int`	**`initialBundleStartLevel`** The initial start level value that is assigned to a bundle when it is first installed.	25
`int`	**`startLevel`** The active start level value for the framework.	25

Constructor Summary	Page
`FrameworkStartLevelDTO``()`	26

Methods inherited from class org.osgi.dto.DTO
`toString`

Field Detail

startLevel

```
public int startLevel
```

The active start level value for the framework.

See Also:

`org.osgi.framework.startlevel.FrameworkStartLevel.getStartLevel()`

initialBundleStartLevel

```
public int initialBundleStartLevel
```

The initial start level value that is assigned to a bundle when it is first installed.

See Also:

`org.osgi.framework.startlevel.FrameworkStartLevel.getInitialBundleStartLevel()`

Constructor Detail

FrameworkStartLevelDTO

```
public FrameworkStartLevelDTO()
```

Package org.osgi.framework.wiring.dto

```
@org.osgi.annotation.versioning.Version(value="1.2")
```

OSGi Data Transfer Object Framework Wiring Package Version 1.2.

See:
Description

Class Summary		Page
BundleRevisionDTO	Data Transfer Object for a BundleWiring.	28
BundleWireDTO	Data Transfer Object for a BundleWire.	30
BundleWiringDTO	Data Transfer Object for a BundleWiring graph.	32
BundleWiringDTO.NodeDTO	Data Transfer Object for a BundleWiring node.	34

Package org.osgi.framework.wiring.dto Description

OSGi Data Transfer Object Framework Wiring Package Version 1.2.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. This package has two types of users: the consumers that use the API in this package and the providers that implement the API in this package.

Example import for consumers using the API in this package:

```
Import-Package: org.osgi.framework.wiring.dto; version="[1.2,2.0)"
```

Example import for providers implementing the API in this package:

```
Import-Package: org.osgi.framework.wiring.dto; version="[1.2,1.3)"
```

Class BundleRevisionDTO

org.osgi.framework.wiring.dto

```
java.lang.Object
  └org.osgi.dto.DTO
      └org.osgi.resource.dto.ResourceDTO
          └org.osgi.framework.wiring.dto.BundleRevisionDTO
```

```
public class BundleRevisionDTO
extends ResourceDTO
```

Data Transfer Object for a BundleWiring.

An installed Bundle can be adapted to provide a `BundleRevisionDTO` for the current revision of the Bundle. `BundleRevisionDTO` objects for all in use revisions of the Bundle can be obtained by adapting the bundle to `BundleRevisionDTO[]`.

NotThreadSafe

Field Summary		Page
long	**bundle** The id of the bundle associated with the bundle revision.	29
String	**symbolicName** The symbolic name of the bundle revision.	28
int	**type** The type of the bundle revision.	29
String	**version** The version of the bundle revision.	29

Fields inherited from class org.osgi.resource.dto.ResourceDTO
capabilities, id, requirements

Constructor Summary	Page
BundleRevisionDTO()	29

Methods inherited from class org.osgi.dto.DTO
toString

Field Detail

symbolicName

```
public String symbolicName
```

The symbolic name of the bundle revision.

See Also:
`org.osgi.framework.wiring.BundleRevision.getSymbolicName()`

type

`public int` **`type`**

The type of the bundle revision.

See Also:
`org.osgi.framework.wiring.BundleRevision.getTypes()`

---

version

`public String` **`version`**

The version of the bundle revision.

See Also:
`org.osgi.framework.wiring.BundleRevision.getVersion()`

---

bundle

`public long` **`bundle`**

The id of the bundle associated with the bundle revision.

See Also:
`org.osgi.framework.BundleReference.getBundle()`

Constructor Detail

BundleRevisionDTO

`public` **`BundleRevisionDTO`**`()`

Class BundleWireDTO

org.osgi.framework.wiring.dto

```
java.lang.Object
  └ org.osgi.dto.DTO
      └ org.osgi.resource.dto.WireDTO
          └ org.osgi.framework.wiring.dto.BundleWireDTO
```

```
public class BundleWireDTO
extends WireDTO
```

Data Transfer Object for a BundleWire.

`BundleWireDTO`s are referenced `BundleWiringDTO.NodeDTO`s.

NotThreadSafe

Field Summary		Page
`int`	**`providerWiring`** The identifier of the provider wiring for the bundle wire.	*30*
`int`	**`requirerWiring`** The identifier of the requiring wiring for the bundle wire.	*30*

Fields inherited from class org.osgi.resource.dto.WireDTO
`capability`, `provider`, `requirement`, `requirer`

Constructor Summary	Page
`BundleWireDTO``()`	*31*

Methods inherited from class org.osgi.dto.DTO
`toString`

Field Detail

providerWiring

```
public int providerWiring
```

The identifier of the provider wiring for the bundle wire.

See Also:
`WiringDTO.id`, `org.osgi.framework.wiring.BundleWire.getProviderWiring()`

---

requirerWiring

```
public int requirerWiring
```

The identifier of the requiring wiring for the bundle wire.

See Also:

`WiringDTO.id`, `org.osgi.framework.wiring.BundleWire.getRequirerWiring()`

Constructor Detail

BundleWireDTO

```
public BundleWireDTO()
```

Class BundleWiringDTO

org.osgi.framework.wiring.dto

```
java.lang.Object
  └org.osgi.dto.DTO
      └org.osgi.framework.wiring.dto.BundleWiringDTO
```

```
public class BundleWiringDTO
extends DTO
```

Data Transfer Object for a BundleWiring graph.

An installed Bundle can be adapted to provide a `BundleWiringDTO` for the current wiring Bundle. `BundleWiringDTO` objects for all in use wirings of the Bundle can be obtained by adapting the bundle to `BundleWiringDTO[]`.

NotThreadSafe

Nested Class Summary		Page
static class	**BundleWiringDTO.NodeDTO** Data Transfer Object for a BundleWiring node.	34

Field Summary		Page
long	**bundle** The id of the bundle associated with the bundle wiring graph.	32
Set<BundleWiringDTO.NodeDTO>	**nodes** The set of wiring nodes referenced by the wiring graph.	33
Set<BundleRevisionDTO>	**resources** The set of resources referenced by the wiring graph.	33
int	**root** The identifier of the root wiring node of the bundle wiring graph.	33

Constructor Summary	Page
BundleWiringDTO()	33

Methods inherited from class org.osgi.dto.DTO
toString

Field Detail

bundle

```
public long bundle
```

The id of the bundle associated with the bundle wiring graph.

See Also:
`org.osgi.framework.BundleReference.getBundle()`

root

```
public int root
```

The identifier of the root wiring node of the bundle wiring graph.

See Also:
WiringDTO.id

---

nodes

```
public Set<BundleWiringDTO.NodeDTO> nodes
```

The set of wiring nodes referenced by the wiring graph.

All wiring nodes referenced by wiring node identifiers in the wiring graph are contained in this set.

---

resources

```
public Set<BundleRevisionDTO> resources
```

The set of resources referenced by the wiring graph.

All resources referenced by resource identifiers in the wiring graph are contained in this set.

Constructor Detail

BundleWiringDTO

```
public BundleWiringDTO()
```

Class BundleWiringDTO.NodeDTO

org.osgi.framework.wiring.dto

```
java.lang.Object
  └org.osgi.dto.DTO
      └org.osgi.resource.dto.WiringDTO
          └org.osgi.framework.wiring.dto.BundleWiringDTO.NodeDTO
```

Enclosing class:

BundleWiringDTO

```
public static class BundleWiringDTO.NodeDTO
extends WiringDTO
```

Data Transfer Object for a BundleWiring node.

The `providedWires` field must contain an array of `BundleWireDTO`s. The `requiredWires` field must contain an array of `BundleWireDTO`s.

NotThreadSafe

Field Summary		Page
boolean	**current** The current state of the bundle wiring.	34
boolean	**inUse** The bundle wiring's in use setting indicates that the bundle wiring is in use.	34

Fields inherited from class org.osgi.resource.dto.WiringDTO
capabilities, id, providedWires, requiredWires, requirements, resource

Constructor Summary	Page
BundleWiringDTO.NodeDTO()	35

Methods inherited from class org.osgi.dto.DTO
toString

Field Detail

inUse

```
public boolean inUse
```

The bundle wiring's in use setting indicates that the bundle wiring is in use.

See Also:

`org.osgi.framework.wiring.BundleWiring.isInUse()`

current

```
public boolean current
```

The current state of the bundle wiring. The bundle wiring's current setting indicates that the bundle wiring is the current bundle wiring for the bundle.

See Also:

`org.osgi.framework.wiring.BundleWiring.isCurrent()`

Constructor Detail

BundleWiringDTO.NodeDTO

```
public BundleWiringDTO.NodeDTO()
```

Package org.osgi.resource.dto

```
@org.osgi.annotation.versioning.Version(value="1.0")
```

OSGi Data Transfer Object Resource Package Version 1.0.

See:
Description

Class Summary		Page
CapabilityDTO	Data Transfer Object for a Capability.	37
CapabilityRefDTO	Data Transfer Object for a reference to a Capability.	39
RequirementDTO	Data Transfer Object for a Requirement.	41
RequirementRefDTO	Data Transfer Object for a reference to a Requirement.	43
ResourceDTO	Data Transfer Object for a Resource.	45
WireDTO	Data Transfer Object for a Wire.	47
WiringDTO	Data Transfer Object for a Wiring node.	49

Package org.osgi.resource.dto Description

OSGi Data Transfer Object Resource Package Version 1.0.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest. This package has two types of users: the consumers that use the API in this package and the providers that implement the API in this package.

Example import for consumers using the API in this package:

```
Import-Package: org.osgi.resource.dto; version="[1.0,2.0)"
```

Example import for providers implementing the API in this package:

```
Import-Package: org.osgi.resource.dto; version="[1.0,1.1)"
```

Class CapabilityDTO

org.osgi.resource.dto

```
java.lang.Object
  └org.osgi.dto.DTO
      └org.osgi.resource.dto.CapabilityDTO
```

```
public class CapabilityDTO
extends DTO
```

Data Transfer Object for a Capability.

NotThreadSafe

Field Summary		Page
Map<String,Object>	**attributes** The attributes for the capability.	38
Map<String,String>	**directives** The directives for the capability.	38
int	**id** The unique identifier of the capability.	37
String	**namespace** The namespace for the capability.	37
int	**resource** The identifier of the resource declaring the capability.	38

Constructor Summary	Page
CapabilityDTO()	38

Methods inherited from class org.osgi.dto.DTO
toString

Field Detail

id

`public int id`

The unique identifier of the capability.

This identifier is transiently assigned and may vary across restarts.

namespace

`public String namespace`

The namespace for the capability.

See Also:
`org.osgi.resource.Capability.getNamespace()`

---

directives

`public Map<String,String> **directives**`

The directives for the capability.

See Also:
`org.osgi.resource.Capability.getDirectives()`

---

attributes

`public Map<String,Object> **attributes**`

The attributes for the capability.

The value type must be a numerical type, Boolean, String, DTO or an array of any of the former.

See Also:
`org.osgi.resource.Capability.getAttributes()`

---

resource

`public int **resource**`

The identifier of the resource declaring the capability.

See Also:
`ResourceDTO.id`, `org.osgi.resource.Capability.getResource()`

Constructor Detail

CapabilityDTO

`public **CapabilityDTO**()`

Class CapabilityRefDTO

org.osgi.resource.dto

```
java.lang.Object
  └org.osgi.dto.DTO
      └org.osgi.resource.dto.CapabilityRefDTO
```

```
public class CapabilityRefDTO
extends DTO
```

Data Transfer Object for a reference to a Capability.

NotThreadSafe

Field Summary		Page
int	**capability** The identifier of the capability in the resource.	*39*
int	**resource** The identifier of the resource declaring the capability.	*39*

Constructor Summary	Page
CapabilityRefDTO()	*40*

Methods inherited from class org.osgi.dto.DTO
toString

Field Detail

capability

`public int capability`

The identifier of the capability in the resource.

See Also:
CapabilityDTO.id

resource

`public int resource`

The identifier of the resource declaring the capability.

See Also:
ResourceDTO.id

Constructor Detail

CapabilityRefDTO

```
public CapabilityRefDTO()
```

Class RequirementDTO

org.osgi.resource.dto

```
java.lang.Object
  └org.osgi.dto.DTO
      └org.osgi.resource.dto.RequirementDTO
```

```
public class RequirementDTO
extends DTO
```

Data Transfer Object for a Requirement.

NotThreadSafe

Field Summary

	Field Summary	Page
Map<String,Object>	**attributes** The attributes for the requirement.	*42*
Map<String,String>	**directives** The directives for the requirement.	*42*
int	**id** The unique identifier of the requirement.	*41*
String	**namespace** The namespace for the requirement.	*41*
int	**resource** The identifier of the resource declaring the requirement.	*42*

Constructor Summary

Constructor Summary	Page
RequirementDTO()	*42*

Methods inherited from class org.osgi.dto.DTO

toString

Field Detail

id

```
public int id
```

The unique identifier of the requirement.

This identifier is transiently assigned and may vary across restarts.

namespace

```
public String namespace
```

The namespace for the requirement.

See Also:
`org.osgi.resource.Requirement.getNamespace()`

directives

`public Map<String,String> **directives**`

The directives for the requirement.

See Also:
`org.osgi.resource.Requirement.getDirectives()`

attributes

`public Map<String,Object> **attributes**`

The attributes for the requirement.

The value type must be a numerical type, Boolean, String, DTO or an array of any of the former.

See Also:
`org.osgi.resource.Requirement.getAttributes()`

resource

`public int **resource**`

The identifier of the resource declaring the requirement.

See Also:
`ResourceDTO.id`, `org.osgi.resource.Requirement.getResource()`

Constructor Detail

RequirementDTO

`public **RequirementDTO**()`

Class RequirementRefDTO

org.osgi.resource.dto

```
java.lang.Object
  └org.osgi.dto.DTO
      └org.osgi.resource.dto.RequirementRefDTO
```

```
public class RequirementRefDTO
extends DTO
```

Data Transfer Object for a reference to a Requirement.

NotThreadSafe

Field Summary		Page
int	**requirement** The identifier of the requirement in the resource.	43
int	**resource** The identifier of the resource declaring the requirement.	43

Constructor Summary	Page
RequirementRefDTO()	44

Methods inherited from class org.osgi.dto.DTO
toString

Field Detail

requirement

```
public int requirement
```

The identifier of the requirement in the resource.

See Also:
RequirementDTO.id

resource

```
public int resource
```

The identifier of the resource declaring the requirement.

See Also:
ResourceDTO.id

Constructor Detail

RequirementRefDTO

```
public RequirementRefDTO()
```

Class ResourceDTO

org.osgi.resource.dto

```
java.lang.Object
  └org.osgi.dto.DTO
      └org.osgi.resource.dto.ResourceDTO
```

Direct Known Subclasses:
BundleRevisionDTO

```
public class ResourceDTO
extends DTO
```

Data Transfer Object for a Resource.

NotThreadSafe

Field Summary		Page
List<CapabilityDTO>	**capabilities** The capabilities of the resource.	45
int	**id** The unique identifier of the resource.	45
List<RequirementDTO>	**requirements** The requirements of the resource.	46

Constructor Summary	Page
ResourceDTO()	46

Methods inherited from class org.osgi.dto.DTO
toString

Field Detail

id

`public int id`

The unique identifier of the resource.

This identifier is transiently assigned and may vary across restarts.

capabilities

`public List<CapabilityDTO> capabilities`

The capabilities of the resource.

See Also:
`org.osgi.resource.Resource.getCapabilities(String)`

requirements

```
public List<RequirementDTO> requirements
```

The requirements of the resource.

See Also:
`org.osgi.resource.Resource.getRequirements(String)`

Constructor Detail

ResourceDTO

```
public ResourceDTO()
```

Class WireDTO

org.osgi.resource.dto

```
java.lang.Object
  └org.osgi.dto.DTO
      └org.osgi.resource.dto.WireDTO
```

Direct Known Subclasses:
BundleWireDTO

```
public class WireDTO
extends DTO
```

Data Transfer Object for a Wire.

NotThreadSafe

Field Summary		Page
CapabilityRefDTO	**capability** Reference to the Capability for the wire.	*47*
int	**provider** The identifier of the provider resource for the wire.	*48*
RequirementRefDTO	**requirement** Reference to the Requirement for the wire.	*47*
int	**requirer** The identifier of the requiring resource for the wire.	*48*

Constructor Summary	Page
WireDTO()	*48*

Methods inherited from class org.osgi.dto.DTO
toString

Field Detail

capability

```
public CapabilityRefDTO capability
```

Reference to the Capability for the wire.

See Also:
`org.osgi.resource.Wire.getCapability()`

---

requirement

```
public RequirementRefDTO requirement
```

Reference to the Requirement for the wire.

See Also:
`org.osgi.resource.Wire.getRequirement()`

provider

`public int provider`

The identifier of the provider resource for the wire.

See Also:
`ResourceDTO.id`, `org.osgi.resource.Wire.getProvider()`

requirer

`public int requirer`

The identifier of the requiring resource for the wire.

See Also:
`ResourceDTO.id`, `org.osgi.resource.Wire.getRequirer()`

Constructor Detail

WireDTO

`public WireDTO()`

Class WiringDTO

org.osgi.resource.dto

```
java.lang.Object
  └org.osgi.dto.DTO
      └org.osgi.resource.dto.WiringDTO
```

Direct Known Subclasses:

BundleWiringDTO.NodeDTO

```
public class WiringDTO
extends DTO
```

Data Transfer Object for a Wiring node.

NotThreadSafe

Field Summary		Page
List<CapabilityRefDTO>	**capabilities** The references to the capabilities for the wiring node.	49
int	**id** The unique identifier of the wiring node.	49
List<WireDTO>	**providedWires** The provided wires for the wiring node.	50
List<WireDTO>	**requiredWires** The required wires for the wiring node.	50
List<RequirementRefDTO>	**requirements** The references to the requirements for the wiring node.	50
int	**resource** The identifier of the resource associated with the wiring node.	50

Constructor Summary	Page
WiringDTO()	50

Methods inherited from class org.osgi.dto.DTO
toString

Field Detail

id

```
public int id
```

The unique identifier of the wiring node.

This identifier is transiently assigned and may vary across restarts.

---

capabilities

```
public List<CapabilityRefDTO> capabilities
```

The references to the capabilities for the wiring node.

See Also:
`org.osgi.resource.Wiring.getResourceCapabilities(String)`

---

requirements

`public List<RequirementRefDTO> requirements`

The references to the requirements for the wiring node.

See Also:
`org.osgi.resource.Wiring.getResourceRequirements(String)`

---

providedWires

`public List<WireDTO> providedWires`

The provided wires for the wiring node.

See Also:
`org.osgi.resource.Wiring.getProvidedResourceWires(String)`

---

requiredWires

`public List<WireDTO> requiredWires`

The required wires for the wiring node.

See Also:
`org.osgi.resource.Wiring.getRequiredResourceWires(String)`

---

resource

`public int resource`

The identifier of the resource associated with the wiring node.

See Also:
`ResourceDTO.id`, `org.osgi.resource.Wiring.getResource()`

Constructor Detail

WiringDTO

`public WiringDTO()`

7 Considered Alternatives

7.1 Compendium DTOs

We decided that RFCs for a given specification should be the place for DTOs for that specification to be defined. The RFC template now has a DTO section. Therefore this RFC will only address the DTOs for the Core specification.

It was discussed how to obtain DTO instances from other services (e.g. ConfigAdmin). An "Adapter" concept was discussed but not agreed to. It was also discussed that the services add new "getDTO" methods. No conclusion was reached.

8 Security Considerations

Data transfer objects have limited behavior by definition. This behavior requires no permissions.

9 Document Support

9.1 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]. Data Transfer Object. https://en.wikipedia.org/wiki/Data_transfer_object

9.2 Author’s Address

Name	BJ Hargrave
Company	IBM Corporation

Name	David Bosschaert
Company	Red Hat

9.3 Acronyms and Abbreviations

DTO – Data Transfer Object

9.4 End of Document