Connect

Draft

27 Pages

Abstract

OSGi Connect provides a mechanism to create and launch an OSGi Framework instance that can install bundles which use content managed outside of the Framework itself. For example, to provide things like resource loading, class loading, bundle entry content and the bundle manifest headers. Among other things this allows for bundles to exist and be installed into the framework from the flat class path, the Java Platform Module System module path, a jlink image, or a native image.

0 Document Information

0.1 License

DISTRIBUTION AND FEEDBACK LICENSE, Version 2.0

The OSGi Alliance hereby grants you a limited copyright license to copy and display this document (the “Distribution”) in any medium without fee or royalty. This Distribution license is exclusively for the purpose of reviewing and providing feedback to the OSGi Alliance. You agree not to modify the Distribution in any way and further agree to not participate in any way in the making of derivative works thereof, other than as a necessary result of reviewing and providing feedback to the Distribution. You also agree to cause this notice, along with the accompanying consent, to be included on all copies (or portions thereof) of the Distribution. The OSGi Alliance also grants you a perpetual, non-exclusive, worldwide, fully paid-up, royalty free, limited license (without the right to sublicense) under any applicable copyrights, to create and/or distribute an implementation of the Distribution that: (i) fully implements the Distribution including all its required interfaces and functionality; (ii) does not modify, subset, superset or otherwise extend the OSGi Name Space, or include any public or protected packages, classes, Java interfaces, fields or methods within the OSGi Name Space other than those required and authorized by the Distribution. An implementation that does not satisfy limitations (i)-(ii) is not considered an implementation of the Distribution, does not receive the benefits of this license, and must not be described as an implementation of the Distribution. "OSGi Name Space" shall mean the public class or interface declarations whose names begin with "org.osgi" or any recognized successors or replacements thereof. The OSGi Alliance expressly reserves all rights not granted pursuant to these limited copyright licenses including termination of the license at will at any time.

EXCEPT FOR THE LIMITED COPYRIGHT LICENSES GRANTED ABOVE, THE OSGi ALLIANCE DOES NOT GRANT, EITHER EXPRESSLY OR IMPLIEDLY, A LICENSE TO ANY INTELLECTUAL PROPERTY IT, OR ANY THIRD PARTIES, OWN OR CONTROL. Title to the copyright in the Distribution will at all times remain with the OSGi Alliance. The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted therein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

THE DISTRIBUTION IS PROVIDED "AS IS," AND THE OSGi ALLIANCE (INCLUDING ANY THIRD PARTIES THAT HAVE CONTRIBUTED TO THE DISTRIBUTION) MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, OR TITLE; THAT THE CONTENTS OF THE DISTRIBUTION ARE SUITABLE FOR ANY PURPOSE; NOR THAT THE IMPLEMENTATION OF SUCH CONTENTS WILL NOT INFRINGE ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.
NEITHER THE OSGi ALLIANCE NOR ANY THIRD PARTY WILL BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATING TO ANY USE OR DISTRIBUTION OF THE DISTRIBUTION.

Implementation of certain elements of this Distribution may be subject to third party intellectual property rights, including without limitation, patent rights (such a third party may or may not be a member of the OSGi Alliance). The OSGi Alliance is not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

The Distribution is a draft. As a result, the final product may change substantially by the time of final publication, and you are cautioned against relying on the content of this Distribution. You are encouraged to update any implementation of the Distribution if and when such Distribution becomes a final specification.

The OSGi Alliance is willing to receive input, suggestions and other feedback (“Feedback”) on the Distribution. By providing such Feedback to the OSGi Alliance, you grant to the OSGi Alliance and all its Members a non-exclusive, non-transferable,

worldwide, perpetual, irrevocable, royalty-free copyright license to copy, publish, license, modify, sublicense or otherwise distribute and exploit your Feedback for any purpose. Likewise, if incorporation of your Feedback would cause an implementation of the Distribution, including as it may be modified, amended, or published at any point in the future ("Future Specification"), to necessarily infringe a patent or patent application that you own or control, you hereby commit to grant to all implementers of such Distribution or Future Specification an irrevocable, worldwide, sublicenseable, royalty free license under such patent or patent application to make, have made, use, sell, offer for sale, import and export products or services that implement such Distribution or Future Specification. You warrant that (a) to the best of your knowledge you have the right to provide this Feedback, and if you are providing Feedback on behalf of a company, you have the rights to provide Feedback on behalf of your company; (b) the Feedback is not confidential to you and does not violate the copyright or trade secret interests of another; and (c) to the best of your knowledge, use of the Feedback would not cause an implementation of the Distribution or a Future Specification to necessarily infringe any third-party patent or patent application known to you. You also acknowledge that the OSGi Alliance is not required to incorporate your Feedback into any version of the Distribution or a Future Specification.

I HEREBY ACKNOWLEDGE AND AGREE TO THE TERMS AND CONDITIONS DELINEATED ABOVE.

0.2 Trademarks

OSGi™ is a trademark, registered trademark, or service mark of the OSGi Alliance in the US and other countries. Java is a trademark, registered trademark, or service mark of Oracle Corporation in the US and other countries. All other trademarks, registered trademarks, or service marks used in this document are the property of their respective owners and are hereby recognized.

0.3 Feedback

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

0.4 Table of Contents

0.5 Terminology and Document Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY" and "OPTIONAL" in this document are to be interpreted as described in 10.1.

`Source code is shown in this typeface.`

0.6 Revision History

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

Revision	Date	Comments
Initial	14 Aug 2019	David Bosschaert, initial content copied from RFP
Update	10 Dec 2019	Thomas Watson, initial content of technical solution

1 Introduction

This RFC discusses the need for an OSGi Framework that can integrate externally defined modules which are outside of the control of the framework. In other words, to connect the OSGi Framework to code and services from the outside world like jars on the class path or the Java Platform Module System (JPMS) module path.

Parts of this work were first explored in RFP-143 OSGiConnect. Although RFP-143 got accepted it never made it into an RFC. This RFC is taking the basic idea and generalizes it to allow for an adjusted set of use cases as well as focusing on a design that could possibly be driven by a launcher that uses a FrameworkFactory to create instances of a Framework that handle bundle content provided outside of the Framework.

2 Application Domain

The OSGi framework consists of a number of layers where the module layer is by far the largest and most complex. This layer has proved to be very useful for large and complex applications that require side by side versioning and encapsulation of their classes.

The OSGi module layer takes care of the class loading for each bundle and isolates bundles in their own class loader, minimizing global space. The consequence of the module layer being responsible for class loading, prevents or at least makes it very difficult to use classes or services that are already present on the class path and whose class space is managed outside of OSGi.

At the same time, functionality running outside of OSGi is not easily able to benefit from the rich service model of OSGi in a standardized way.

Finally, the OSGi framework assumes it is in control of the actual deployment units i.e., the jar files that get installed as Bundles. There is no way to represent outside content as bundles where the framework hasn't been given the deployment unit.

2.1 Terminology + Abbreviations

OSGi Connect – Working name given to this effort.

Connect Content – Provides a Framework access to content from outside the Framework that can be used to represent an installed bundle in the Framework. A connect content provides things like the class loader for the bundle, access to entries in a bundle (e.g. reading serviced component XML), and the bundle manifest headers. A bundle representing connect content might have some limitations with respect to class loading and isolation.

Connect Module – Provides the current connect content available for a bundle installed or updated in the Framework. If the connect content for a bundle is constant then the connect module may return the same connect content instance for the lifetime of the Framework.

Connect Factory – Hooks into the initialization of the Framework and provides a lookup of connect module instances for bundles installed in the Framework. TODO – I don't prefer the Factory term here because this is more about looking up existing Connect Module instances than creating new ones. Maybe rename to ConnectModules (plural)?

Connect Bundle – A bundle installed in the Framework that has its content being provided by a Connect Content.

3 Problem Description

Code running outside of the OSGi framework is hard to use from the inside. The framework can delegate some of it via the system bundle exports but for "normal" Java applications that is often not enough and somewhat tedious in any case as the delegation is rather static. Furthermore, it isn't possible to represent logical units on the outside (like JPMS Modules, OSGi bundles on the classpath, or other components) inside the framework (as the only delegation method is packages exposed via the system bundle).

That makes it hard and in some cases impossible to create hybrid solutions that can be used inside OSGi as well as in other contexts and that would still work if the two are used within the same application. In other words, there is a need for a solution that allows to bridge part of the outside world into the OSGi framework in such a way that normal OSGi mechanisms apply to it.

4 Requirements

4.1 Basic

- BA0010 – It must be possible to install Connect Bundles.
- BA0020 – It must be possible for Connect bundles to participate in the Service Layer as usual for Bundles.
- BA0030 – It must be possible to use OSGi services implementing an API provided by a Connect Bundle outside of the OSGi framework.

4.2 Connect Bundle

- CB0010 – It must be possible to provide the class loader for a connect bundle.
- CB0020 – It must be possible to influence the wiring of a connect bundle similar to existing resolver hooks.
- CB0030 – It must be possible to provide the manifest headers of a connect bundle.
- CB0040 – It must be possible to provide the resources/entries of a connect bundle.
- CB0050 – Connect bundles must be treated and behave like any other bundle except for a limited set of functionalities that are explicitly not supported.
- CB0060 – Connect Bundles must be subject to the standard OSGi resolution rules.
- CB0070 – All existing lifecycle methods must be well-defined wrt. Connect Bundles.
- CB0080 – Installed Connect Bundles must be able to be persisted across restarts.

5 Technical Solution

A new package `org.osgi.framework.connect` is defined that has the Connect API. The `org.osgi.framework.launch.FrameworkFactory` interface is extended to include the following method:

```
Framework newFramework(Map<String,String> configuration,
        ConnectFactory connectFactory);
```

This method will create a new Framework instance that uses the specified `ConnectFactory`.

5.1 Storage

If the framework supports persistence then the framework determines the path used for storage area according to the launch property `org.osgi.framework.storage`. Once the framework instance has determined the storage area it must call the `ConnectFactory` method:

```
void initialize(File storage, Map<String,String> configuration);
```

The storage `File` is the storage area used by the Framework and may be `null` if persistence is not supported. The config `Map` is the unmodifiable map of framework configuration properties that were used to create the new Framework instance.

The `ConnectFactory initialize` method is called once for the life time of the framework instance and it must be called before any other method on the `ConnectFactory`. If the framework instance is stopped and started again the `ConnectFactory initialize` method is not called again because it is not possible to change the storage or the configuration without constructing a new framework instance.

5.2 Activation

A `ConnectFactory` may hook into the lifecycle of the framework itself. This is done by the `ConnectFactory` providing a `BundleActivator` instance.

5.2.1 Start

When the framework is initialized the system bundle enters the STARTING state. At this point a valid `BundleContext` exists for the framework. Before calling any extension bundle activator `start` methods the framework must call the `ConnectFactory` method:

```
Optional<BundleActivator> createBundleActivator();
```

If the `ConnectFactory` provides a bundle activator then its `start` method must be called before any extension bundle activator `start` methods are called and before returning from `Framework.init` method.

The activator allows for the `ConnectFactory` hook into the lifecycle of the framework itself. For example, this allows the `ConnectFactory` to register services, add listeners, install other bundles etc. before other bundles installed in the framework do, including framework extension bundles. This is important to allow for a `ConnectFactory` to influence behavior of the framework by registering various framework hooks like a `org.osgi.framework.hooks.resolver.ResolverHook`. A `ResolverHook` is useful for cases where it does not make sense to allow the wiring of a connect bundle to to wire to capabilities provided by a normal bundle installed in the framework. In that case the `ConnectFactory` can register a `ResolverHook` to limit what a connect bundle wiring gets wired to.

5.3 Shutdown

When the framework is stopped it eventually reaches start level 0 and the framework checks to see if there are any framework extension activators to call the `stop` method on. After calling `stop` on framework extension activators the framework must call `stop` on the activator provided by the `ConnectFactory` at framework initialization.

5.4 Connect Content Install

When a bundle is installed a bundle location and optionally an input stream to the content is provided to the framework. Before the framework reads the content of the bundle, the framework must call the `ConnectFactory` method:

```
Optional<ConnectModule> getModule(String location);
```

The `getModule` method is given the bundle location used to install the bundle. The `getModule` method must do one of the following:

1. Throw an `IllegalStateException` if the installation of the bundle is to be prevented. In this case a `BundleException` is thrown from install with the cause of the `IllegalStateException`. Any other unchecked exception thrown by the `getModule` method must also result in a `BundleException`.

2. Return an empty `Optional` indicating that the Framework must handle reading the content of the bundle itself.

3. Return a present `Optional` indicating that the `ConnectModule` present must be used to access the content of the bundle.

If a `ConnectModule` is found for the bundle location then the following method is called by the Framework to get the current `ConnectContent` for the bundle:

```
ConnectContent getContent() throws IOException;
```

If an `IOException` is thrown it must result in a `BundleException` with the thrown `IOException` as the cause. Any other unchecked exception thrown by the `getContent` method must also result in a `BundleException`.

The `ConnectContent` is then used by the framework to access content of the bundle's current `BundleRevision`.

5.5 Connect Content Update

If one of the `Bundle.update` methods is called for a bundle that has a `ConnectModule` present for the bundle location then the `ConnectModule.getContent` method must be called to get the current `ConnectContent` for the bundle. A `ConnectModule` is allowed to return the same `ConnectContent` or a different one each time `getContent` is called.

If an `IOException` is thrown it must result in a `BundleException` with the thrown `IOException` as the cause. Any other unchecked exception thrown by the `getContent` method must also result in a `BundleException`.

The `ConnectContent` is then used by the framework to access content of the bundle's current `BundleRevision` after the bundle update.

5.6 Reading Connect Content

The `ConnectContent` provides the framework with all the information and resources necessary to represent a `BundleRevision` in the framework.

5.6.1 Opening Connect Content

Before accessing the `ConnectContent` a framework must first open the `ConnectContent` with the following method:

```
void open() throws IOException;
```

If an IOException is thrown while opening the content during install or update then it must result in a `BundleException` with the thrown `IOException` as the cause. A framework may open and close the content many times while the bundle is installed in the framework. For example, to limit the number of resources kept open concurrently by the framework. The framework must always ensure that the `ConnectContent` is open before calling other methods on the `ConnectContent`.

5.6.2 Bundle Manifest Headers

A `ConnectContent` may provide the bundle manifest headers to be used for the bundle. The framework must call the following method:

```
Optional<Map<String,String>> getHeaders();
```

If an empty `Optional` is returned then the framework must parse the bundle manifest itself by accessing the `META-INF/MANIFEST.MF` entry of the `ConnectContent`.

If the `Optional` returned has a `Map` present then the `Map` must be used to provide the bundle headers. The headers must be used the same way the header values would have been used from a parsed `META-INF/MANIFEST.MF` entry. That is the header keys and values that have semantic meaning must be used by the framework for the bundle and the key/value pairs must be used for the `Dictionary` returned by `Bundle.getHeaders` method. The `BundleRevision` associated with the `ConnectContent` must also have its capabilities and requirements defined by the contents of the `Map`.

5.6.3 Bundle Class Loader

A `ConnectContent` may provide the class loader to use for a bundle. Before creating a class loader for a bundle using a `ConnectContent` the framework must call the following method:

```
Optional<ClassLoader> getClassLoader();
```

If an empty `Optional` is returned then the framework must create a class loader for the bundle. The class loader created by the framework must follow all the delegation rules defined by the OSGi module layer for a bundle class loader and it must implement the `BundleReference` interface. All resources found and classes defined by this class loader must have their content read using the `ConnectContent` entries. This is similar to how a bundle class loader works when the framework is responsible for ready bundle JAR files directly.

If the `Optional` returned has a `ClassLoader` present then that `ClassLoader` must be used as the class loader for the `BundleWiring` associated with the `ConnectContent`. The `ClassLoader` provided by the `ConnectContent` is not required to implement the `BundleReference` interface and is not required to follow the delegation rules defined by the OSGi module layer.

A `ConnectFactory` is not required to provide a unique class loader for each `ConnectContent`. That is the same class loader can be used as the class loader for multiple `ConnectContent` objects and therefore get used by multiple bundles installed in the framework. If the connect bundle exports packages then the `ConnectContent` class loader will be delegated to by other class loaders managed by the framework which do not have an associated `ConnectContent`. In other words, a bundle installed with no `ConnectContent` associated with it may import packages from connect bundles and the OSGi module layer will do the correct delegation of class loads to the connect class loader as defined by the OSGi module layer for exported packages.

5.6.4 Connect Content Entries

A `ConnectContent` provides access to connect entries to be used by the framework for two purposes:

1. To provide content for the framework `Bundle` and `BundleWiring` methods which introspect bundle entries. For example, the `Bundle.getEntry`, `Bundle.getEntryPaths` and `BundleWiring.findEntries` methods.

2. To provide content when the framework is responsible for creating the class loader for the module.

A resource in a `ConnectContent` is identified by a path name that is a '/'-separated path string. A `ConnectContent` may treat directories as resources, but it is not required to. Where a `ConnectContent` does contain a directory that can be located then its path name must end with a slash ('/'). The directory can also be located with a path name that drops the trailing slash.

5.6.4.1 Discover Connect Entry Names

All `ConnectEntry` names provided by a `ConnectContent` can be introspected by the `ConnectContent` method:

```
Iterable<String> getEntries() throws IOException;
```

The iterable provided is used to by the framework for both the `Bundle.getEntryPaths` and `BundleWiring.findEntries` methods.

5.6.4.2 Connect Entry Lookup

The content of an entry contained in a `ConnectContent` can be introspected using the `ConnectContent.ConnectEntry` interface. A `ConnectEntry` can be found by its path name using the `ConnectContent` method:

```
Optional<ConnectEntry> getEntry(String path);
```

An empty `Optional` is returned if the entry does not exist with the specified path name. Otherwise the present `ConnectEntry` is returned. The `ConnectEntry` can be used by the framework to create URLs with the framework specific protocol for bundle entries. For example, the URLs returned by the `Bundle.getEntry` and `BundleWiring.findEntries` methods.

The framework must ensure that the path value used does not start with slash ('/'). That is any paths used for `Bundle.getEntry` and `BundleWiring.findEntries` must have the beginning slash removed before calling `ConnectContent.getEntry`

5.6.5 Closing Connect Content

Once a `BundleRevision` which is backed by a `ConnectContent` is no longer in use by the framework then the framework must close the `ConnectContent` with the method:

```
void close() throws IOException;
```

The framework is free to close the `ConnectContent` at any other time during the lifecycle of the framework, but it must always ensure `ConnectContent` is opened before calling other methods on it.

5.7 Launching From Persistent Storage

The framework must make a record of which bundles are installed using a `ConnectModule`. When the framework is stopped it must persist the state of all the installed bundles, including the ones using a `ConnectModule`. When a new framework instance is created using the persistent storage which recorded the use of a `ConnectModule`, the framework must verify that a `ConnectFactory` is available that can handle the bundle location.

If there is no `ConnectModule` present then the bundle installed must be discarded by the framework as if it is not installed and a warning framework event should be published or a warning should be logged.

5.8 FrameworkUtil Helper

The class `FrameworkUtil` has a number of static utility methods. The following method may be of interest to connect factory implementations:

```
public static Bundle getBundle(final Class<?> classFromBundle)
```

The classes from connect bundles, depending on if their class loader implements `BundleReference`, may return a `null` bundle from the `getBundle` method. To allow a connect factory implementation to hook into the `FrameworkUtil` class a new interface exists in the `org.osgi.framework.connect` package:

```
public interface FrameworkUtilHelper {

	default Bundle getBundle(Class< ? > classFromBundle) {
		return null;
	}
}
```

The `FrameworkUtil` class has been enhanced to to get a list of available `FrameworkUtilHelper` implementations using the `ServiceLoader`. The `FrameworkUtil.getBundle` method has also been enhanced to fall back to using the available `FrameworkUtilHelper` implementations when the class loader of the `classFromBundle` is not a `BundleReference`.

The immutable list of available `FrameworkUtilHelper` is determined during class initialization of the `FrameworkUtil` class and cannot change during the lifetime of the `FrameworkUtil` class.

6 Data Transfer Objects

No new DTOs required for this RFC

7 Javadoc

OSGi Javadoc

12/9/19 3:35 PM

Package Summary		Page
org.osgi.framework.connect	Framework Connect Package Version 1.0.	14
org.osgi.framework.launch	Framework Launch Package Version 1.2.	24

Package org.osgi.framework.connect

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

Framework Connect Package Version 1.0.

See:

Description

Interface Summary		Page
ConnectContent	A connect content provides a `framework` access to the content of a connect `module`.	15
ConnectContent.ConnectEntry	Represents the entry of a connect module	18
ConnectFactory	A connect factory creates instances of `ConnectModule` that are used by a `org.osgi.framework.launch.Framework` instance to provide content and classes for a bundle installed in the Framework.	20
ConnectModule	A connect module instance is used by a `framework` to load content for a bundle revision installed in the framework.	22
FrameworkUtilHelper	A helper for the `org.osgi.framework.FrameworkUtil` class.	23

Package org.osgi.framework.connect Description

Framework Connect Package Version 1.0.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest.

Example import for consumers using the API in this package:

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

Interface ConnectContent

org.osgi.framework.connect

```
public interface ConnectContent
```

A connect content provides a `framework` access to the content of a connect `module`. A framework may `open` and `close` the content for a connect module multiple times while the connect content is in use by the framework instance. The framework must close the connect content once the connect content is no longer used as the content of a current bundle revision or an in use bundle revision.

See Also:
`org.osgi.framework.wiring.BundleRevisions`

ThreadSafe

Nested Class Summary		Page
static interface	**ConnectContent.ConnectEntry** Represents the entry of a connect module	18

Method Summary		Page
void	**close**() Closes this connect content.	17
Optional<ClassLoader>	**getClassLoader**() Returns a class loader for this connect content.	16
Iterable<String>	**getEntries**() Returns an iterable with all the entry names available in this ConnectContent	16
Optional<ConnectContent.ConnectEntry>	**getEntry**(String path) Returns the connect entry for the specified path name in this content.	16
Optional<Map<String, String>>	**getHeaders**() Returns this connect content Manifest headers and values.	15
void	**open**() Opens this connect content.	16

Method Detail

getHeaders

```
Optional<Map<String,String>> getHeaders()
```

Returns this connect content Manifest headers and values. The `empty` value is returned if the framework should handle parsing the Manifest of the content itself.

Returns:
This connect content Manifest headers and values.

Throws:
`IllegalStateException` - if the connect content has been closed

---

getEntries

```
Iterable<String> getEntries()
                    throws IOException
```

Returns an iterable with all the entry names available in this ConnectContent

Returns:
the entry names
Throws:
`IOException` - if an error occurs reading the ConnectContent
`IllegalStateException` - if the connect content has been closed

---

getEntry

```
Optional<ConnectContent.ConnectEntry> getEntry(String path)
```

Returns the connect entry for the specified path name in this content. The `empty` value is returned if an entry with the specified path name does not exist. The path must not start with a "/" and is relative to the root of this content.

Parameters:
`path` - the path name of the entry
Returns:
the connect entry, or `empty` if not found.
Throws:
`IllegalStateException` - if the connect content has been closed

---

getClassLoader

```
Optional<ClassLoader> getClassLoader()
```

Returns a class loader for this connect content. The `empty` value is returned if the framework should handle creating a class loader for the bundle revision associated with this connect content.

This method is called by the framework for `resolved` bundles only and will be called at most once while a bundle is resolved. If a bundle associated with a connect module is refreshed and resolved again the framework will ask the content for the class loader again. This allows for a connect content to reuse or create a new class loader each time the bundle revision is resolved.

Returns:
a class loader for the module.

---

open

```
void open()
        throws IOException
```

Opens this connect content. The framework will open the content when it needs to access the content for a bundle revision associated with the connect content. The framework may lazily open the content until the first request is made to access the bundle revision content.

Throws:
`IOException` - if an error occurred opening the content

---

close

```
void close()
    throws IOException
```

Closes this connect content.

Throws:

`IOException` - if an error occurred closing the connect content

Interface ConnectContent.ConnectEntry

org.osgi.framework.connect

Enclosing class:
ConnectContent

```
public static interface ConnectContent.ConnectEntry
```

Represents the entry of a connect module

Method Summary		Page
byte[]	**getBytes**() Returns the content of the entry as a byte array.	19
long	**getContentLength**() Returns the size of the entry.	18
InputStream	**getInputStream**() Returns the content of the entry as an input stream.	19
long	**getLastModified**() Returns the last modification time of the entry	18
String	**getName**() Returns the path name of the entry	18

Method Detail

getName

```
String getName()
```

Returns the path name of the entry

Returns:
the path name of the entry

---

getContentLength

```
long getContentLength()
```

Returns the size of the entry. The value `-1` is returned if the content length is not known.

Returns:
the size of the entry, or `-1` if the content length is not known.

---

getLastModified

```
long getLastModified()
```

Returns the last modification time of the entry

Returns:
the last modification time of the entry

---

getBytes

```
byte[] getBytes()
         throws IOException
```

Returns the content of the entry as a byte array.

Returns:
the content bytes
Throws:
`IOException` - if an error occurs reading the content

---

getInputStream

```
InputStream getInputStream()
                    throws IOException
```

Returns the content of the entry as an input stream.

Returns:
the content input stream
Throws:
`IOException` - if an error occurs reading the content

Interface ConnectFactory

org.osgi.framework.connect

`public interface` **`ConnectFactory`**

A connect factory creates instances of `ConnectModule` that are used by a `org.osgi.framework.launch.Framework` instance to provide content and classes for a bundle installed in the Framework. A connect factory is provided when `creating` a framework instance. Because a connect factory instance can participate in the initialization of the framework and the lifecycle of a framework instance the connect factory instance should only be used with a single framework instance.

ThreadSafe

Method Summary		Page
`Optional<org.osgi.framework.BundleActivator>`	**`createBundleActivator`**`()` Creates a new activator for this factory.	21
`Optional<ConnectModule>`	**`getModule`**`(String location)` Returns the connect module for the specified bundle location.	20
`void`	**`initialize`**`(File storage, Map<String,String> configuration)` Initializes the connect factory with the `framework persistent storage` file and framework properties configured for a `org.osgi.framework.launch.Framework` instance.	20

Method Detail

initialize

```
void initialize(File storage,
                Map<String,String> configuration)
```

Initializes the connect factory with the `framework persistent storage` file and framework properties configured for a `org.osgi.framework.launch.Framework` instance. This method is called once by a `org.osgi.framework.launch.Framework` instance and is called before any other methods on this factory are called.

Parameters:

`storage` - the persistent storage area used by the `org.osgi.framework.launch.Framework` or `null` if the if the platform does not have file system support.

`configuration` - The framework properties to used configure the new framework instance. An unmodifiable map of framework configuration properties that were used to create a new framework instance.

---

getModule

```
Optional<ConnectModule> getModule(String location)
```

Returns the connect module for the specified bundle location. If an `empty` optional is returned the the framework must handle reading the content of the bundle itself. If a value is `present` in the returned optional then the `value` from the optional must be used to access the content of the bundle.

Parameters:

`location` - the bundle location used to install a bundle

Returns:
the connect module for the specified bundle location
Throws:
`IllegalStateException` - if the location cannot be handled

---

createBundleActivator

`Optional<org.osgi.framework.BundleActivator>` **`createBundleActivator`**`()`

Creates a new activator for this factory. A new activator is created by the framework each time the framework is `initialized`. An activator allows the factory to participate in the framework lifecyle. When the framework is `initialized` the activator `start` method is called. When the framework is `stopped` the activator `stop` method is called

Returns:
a new activator for this factory or `empty` if no activator is available for the factory

Interface ConnectModule

org.osgi.framework.connect

```
public interface ConnectModule
```

A connect module instance is used by a `framework` to load content for a bundle revision installed in the framework.

ThreadSafe

Method Summary		Page
ConnectContent	**getContent**() Returns the current content of this connect module.	22

Method Detail

getContent

```
ConnectContent getContent()
                    throws IOException
```

Returns the current content of this connect module. The framework will get the content when it needs to access the content for the current `bundle revision` associated with this connect module. The framework may lazily open the content until the first request is made to access the bundle content.

Returns:
the current content of this connect module

Throws:
`IOException` - if an error occurred getting the content

Interface FrameworkUtilHelper

org.osgi.framework.connect

`public interface` **`FrameworkUtilHelper`**

A helper for the `org.osgi.framework.FrameworkUtil` class. This helper provides alternative implementations for methods on `org.osgi.framework.FrameworkUtil`.

Method Summary		Page
`org.osgi.framework.Bundle`	**`getBundle`**`(Class<?> classFromBundle)` Return a `Bundle` associated with the specified class.	23

Method Detail

getBundle

`org.osgi.framework.Bundle` **`getBundle`**`(Class<?> classFromBundle)`

Return a `Bundle` associated with the specified class.

This helper method is called by `org.osgi.framework.FrameworkUtil.getBundle(Class)` if the standard implementation of `FrameworkUtil` cannot find the bundle.

Parameters:

`classFromBundle` - A class associated with a bundle

Returns:

A `Bundle` for the specified class or `null` if the specified class is not from a bundle.

Package org.osgi.framework.launch

```
@org.osgi.annotation.versioning.Version(value="1.3")
```

Framework Launch Package Version 1.2.

See:

Description

Interface Summary		Page
FrameworkFactory	A factory for creating `org.osgi.framework.launch.Framework` instances.	25

Package org.osgi.framework.launch Description

Framework Launch Package Version 1.2.

Bundles wishing to use this package must list the package in the Import-Package header of the bundle's manifest.

Example import for consumers using the API in this package:

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

Interface FrameworkFactory

org.osgi.framework.launch

```
@org.osgi.annotation.versioning.ProviderType
public interface FrameworkFactory
```

A factory for creating `org.osgi.framework.launch.Framework` instances.

A framework implementation jar must contain the following resource:

```
/META-INF/services/org.osgi.framework.launch.FrameworkFactory
```

This UTF-8 encoded resource must contain the name of the framework implementation's FrameworkFactory implementation class. Space and tab characters, including blank lines, in the resource must be ignored. The number sign (`'#'` \u0023) and all characters following it on each line are a comment and must be ignored.

Launchers can find the name of the FrameworkFactory implementation class in the resource and then load and construct a FrameworkFactory object for the framework implementation. The FrameworkFactory implementation class must have a public, no-argument constructor. Java™ SE 6 introduced the `ServiceLoader` class which can create a FrameworkFactory instance from the resource.

ThreadSafe

Method Summary		Page
`org.osgi.framework.launch.Framework`	**newFramework**(Map<String,String> configuration) Create a new `org.osgi.framework.launch.Framework` instance.	25
`org.osgi.framework.launch.Framework`	**newFramework**(Map<String,String> configuration, ConnectFactory connectFactory) Create a new `org.osgi.framework.launch.Framework` instance using the specified connect factory.	26

Method Detail

newFramework

```
org.osgi.framework.launch.Framework newFramework(Map<String,String> configuration)
```

Create a new `org.osgi.framework.launch.Framework` instance.

Parameters:

`configuration` - The framework properties to configure the new framework instance. If framework properties are not provided by the configuration argument, the created framework instance must use some reasonable default configuration appropriate for the current VM. For example, the system packages for the current execution environment should be properly exported. The specified configuration argument may be `null`. The created framework instance must copy any information needed from the specified configuration argument since the configuration argument can be changed after the framework instance has been created.

Returns:

A new, configured `org.osgi.framework.launch.Framework` instance. The framework instance must be in the `org.osgi.framework.Bundle.INSTALLED` state.

Throws:

`SecurityException` - If the caller does not have `AllPermission`, and the Java Runtime Environment supports permissions.

newFramework

```
org.osgi.framework.launch.Framework newFramework(Map<String,String> configuration,
                                                 ConnectFactory connectFactory)
```

Create a new `org.osgi.framework.launch.Framework` instance using the specified `connect factory`.

Parameters:

`configuration` - The framework properties to configure the new framework instance. If framework properties are not provided by the configuration argument, the created framework instance must use some reasonable default configuration appropriate for the current VM. For example, the system packages for the current execution environment should be properly exported. The specified configuration argument may be `null`. The created framework instance must copy any information needed from the specified configuration argument since the configuration argument can be changed after the framework instance has been created.

`connectFactory` - The connect factory that the new framework instance will use. The specified connect factory argument may be `null`.

Returns:

A new, configured `org.osgi.framework.launch.Framework` instance. The framework instance must be in the `org.osgi.framework.Bundle.INSTALLED` state.

Throws:

`SecurityException` - If the caller does not have `AllPermission`, and the Java Runtime Environment supports permissions.

Since:

1.3

See Also:

`ConnectFactory`

8 Considered Alternatives

In the early days of the Connect concept the idea was to produce a trimmed down Framework implementation that only understood how to load static things from the class path. An implementation of this idea was done in the Apache Felix project called PojoSR.

This approach is not used because may require forking the framework implementation for each Connect scenario. A better option is to add standard SPI to launch a standard Framework which allows for the Connect scenarios to work with compliant Framework implementations without modifying the Framework implementation itself.

9 Security Considerations

Connect content is managed from outside the Framework. The class loader used to load classes may not be provided by the Framework itself. The protection domain used for the classes from connect content also may not be provided by the Framework. This limits the ability to assign permissions to the classes from the connect content.

A Connect Factory is passed the bundle location of all bundles installed in the framework. The bundle location may contain sensitive. Any Connect Factory uses must be trusted to handle such information in a secure way.

10 Document Support

10.1 References

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

[2]. Software Requirements & Specifications. Michael Jackson. ISBN 0-201-87712-0

10.2 Author's Address

Name	
Company	
Address	
Voice	
e-mail	

10.3 Acronyms and Abbreviations

10.4 End of Document