

OSGi Service Adapter Reference Guide



April 2007

This edition applies to the IBM Unstructured Information Management Architecture (UIMA) SDK Version 1.4.2 and to all subsequent release and modifications until otherwise indicated in new editions.

© Copyright International Business Machines Corporation 2004, 2007. All rights reserved.

US Government Users Restricted Rights - Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

CONTENTS

1	General architecture and responsibilities.....	5
2	OSGi Service Adaptor modules	5
3	OSGi Service Adaptor API	7
3.1	UIMA Services Integrator.....	7
3.2	UIMA Configurator	9
3.3	UIMA Component Services	11

1 General architecture and responsibilities

OSGi Service Adaptor represents a layer between UIMA Java Framework and OSGi Framework that allows UIMA applications to work with UIMA analytics and other resources encapsulated in OSGi bundles and deployed in OSGi container.

General architecture of the OSGi Service Adaptor layer is depicted in Figure 1.

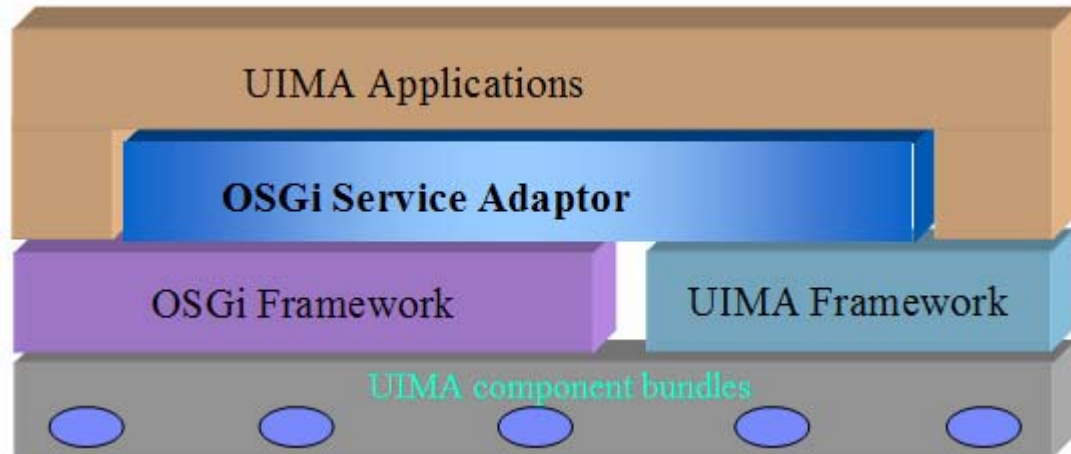


Figure 1. General architecture of OSGi Service Adaptor layer.

The OSGi Service Adaptor is responsible for:

- Allowing UIMA applications to manage OSGi bundles encapsulating UIMA analytics and other resources; the management capabilities include locating UIMA component bundles and loading them into the OSGi container to make them available for deployment by UIMA Framework.
- Providing UIMA application with easy access to shared resources of UIMA component bundles in order to deploy and use them in UIMA Framework; the shared resources include component descriptors, component class space, metadata and others.

2 OSGi Service Adaptor modules

The OSGi Service Adaptor software includes the following 2 bundles:

- `com.ibm.uima.osgi.service` – defines the Service Adaptor API; this bundle is used by *active* UIMA component bundles as well as UIMA applications;
- `com.ibm.uima.osgi.service.internal` – implements most of the Service Adaptor modules; this bundle also refers to the API bundle.

Logically, the OSGi Service Adaptor comprises the following functional modules:

- UIMA Services Integrator – responsible for tracking all standard Adaptor services and notifying applications of UIMA service availability events. This module also provides convenient access to all standard Adaptor services.
- UIMA Configurator – responsible for managing UIMA component bundles and notifying applications of UIMA component life cycle events.
- UIMA Component Services – responsible for providing access to shared resources of UIMA component bundles, loaded into the OSGi container.
- UIMA Component Registry – responsible for looking-up UIMA component bundles loaded into the OSGi container.

The UIMA Services Integrator module's API and implementation code is in the `com.ibm.uima.osgi.service` bundle. The API of all the other functional modules is declared in the `com.ibm.uima.osgi.service` bundle and the implementation code is in the `com.ibm.uima.osgi.service.internal` bundle.

The UIMA Services Integrator module monitors all standard Adaptor services by using the OSGi `ServiceTracker` capabilities and makes its own OSGi service available only when all other standard Adaptor services are available. The Integrator module also monitors its own OSGi service and notifies subscribed applications of the UIMA service availability events.

The UIMA Configurator module combines several functions:

- discovering new UIMA component bundle files in the designated directory;
- installing UIMA component bundles in the OSGi container upon application request;
- automatically activating UIMA components installed in the OSGi container;
- monitoring UIMA component life cycle changes and notifying subscribed applications of UIMA component events.

The module registers its own OSGi service for installing UIMA component bundles and receiving UIMA component events.

The UIMA Component Services module defines the API for accessing shared resources of UIMA component bundles, including:

- component bundle class loader – the bundle class loader established by OSGi container;
- component UID – a unique identifier of a component bundle;
- component metadata, such as symbolic name, version and properties;

- component descriptor(s), called specifications; each component may define one or more UIMA descriptors (specifications) in its OSGi bundle manifest.

This module also provides the API for *publishing* UIMA components, i.e. registering OSGi services that belong to UIMA component bundles. Each *active* UIMA component bundle registers two or more OSGi services at the start-up: (1) component reference service, providing access to component bundle resources, and (2) component specification service(s) for each UIMA component descriptor defined in the bundle manifest. In addition, the UIMA Component Services module registers the UIMA Component Publisher OSGi service for publishing UIMA component bundles.

The UIMA Component Registry module defines the API for looking-up UIMA component services in the OSGi Service Registry. The Registry allows to look-up both UIMA component references and specifications based on internal bundle ID, component UID and its metadata.

3 OSGi Service Adaptor API

Formal description of the API packages, classes and methods can be found in the ‘Help/Help Contents/OSGi UIMA Documentation’ menu in your Eclipse IDE or online at <http://imsgroup03.watson.ibm.com/UIMA-OSGi/api/doc/>.

3.1 UIMA Services Integrator

Interface `com.ibm.uima.osgi.service.IUIMAServices`.

This is the interface of the UIMA Services Integrator, registered as OSGi service by the `com.ibm.uima.osgi.service` bundle. This service provides access to all standard services registered by the OSGi Service Adaptor.

Methods:

- `getConfigurator()` – returns the reference to the UIMA Configurator service, if the service is available, `null` otherwise;
- `getRegistry()` – returns the reference to the UIMA Component Registry service, if the service is available, `null` otherwise;
- `publishComponent()` – creates and registers component services for the given UIMA component bundle.

Class `com.ibm.uima.osgi.service.UIMAServices`.

This class provides convenient static access to all standard services registered by the OSGi Service Adaptor and allows getting notifications of UIMA service events.

Static methods:

- `getInstance()` - returns the single instance of this class;

- `getServices()` – returns the reference to the UIMA Services Integrator service, if the service is available, null otherwise;
- `getConfigurator()` - returns the reference to the UIMA Configurator service, if the service is available, null otherwise;
- `getRegistry()` - returns the reference to the UIMA Component Registry service, if the service is available, null otherwise;
- `publishComponent()` - creates and registers component services for the given UIMA component bundle;
- `addServiceListener()` – adds the given listener object to receive notifications of UIMA service events;
- `removeServiceListener()` – removes the given listener object from the list;
- `newComponentUID()` – creates new UID object for the given component;
- `newComponentSpecUID()` – creates new UID object for the given component specification.

Interface `com.ibm.uima.osgi.service.UIMAServiceListener`.

This interface should be implemented by applications that need to receive notifications of UIMA service events. Applications can subscribe for receiving UIMA service events through the `addServiceListener()` static method of the `UIMAServices` class, and unsubscribe through the `removeServiceListener()` static method of the `UIMAServices` class.

Methods:

- `serviceChanged()` – this method is called by the UIMA Services Integrator service to notify the subscribed application of the given UIMA service event.

Class `com.ibm.uima.osgi.service.event.UIMAServiceEvent`.

This class encapsulates the attributes of a given UIMA service event. UIMA service event objects are delivered to subscribed listeners when a change occurs in one of the OSGi Service Adaptor service's lifecycle. The service code identifies the OSGi Service Adaptor service, and the type code identifies the event type.

Service codes:

- `UIMA_COMPONENT_PUBLISHER` - identifies UIMA Component Publisher service;
- `UIMA_CONFIGURATOR` - identifies UIMA Configurator service;
- `UIMA_REGISTRY` - identifies UIMA Component Registry service;
- `UIMA_SERVICES` - identifies UIMA Services Integrator service.

Type codes:

- `AVAILABLE` - indicates that the service becomes available;

- `MODIFIED` - indicates that the service has been modified;
- `UNAVAILABLE` - indicates that the service becomes unavailable.
- *Methods:*
- `getServiceCode()` – returns the specified event service code;
- `getType()` – returns the specified event type code.

3.2 UIMA Configurator

Interface `com.ibm.uima.osgi.service.IUIMAConfigurator`.

This is the interface of the UIMA Configurator, registered as OSGi service by the `com.ibm.uima.osgi.service.internal` bundle.

Methods:

- `installBundle()` - installs a bundle from a given file in the OSGi container;
- `addComponentListener()` – adds the given listener object to receive notifications of UIMA component events;
- `removeComponentListener()` – removes the given listener object from the list.

Interface `com.ibm.uima.osgi.service.IUIMAComponentListener`.

This interface should be implemented by applications that need to receive notifications of UIMA component events. Applications can subscribe for receiving UIMA component events through the `addComponentListener()` method of the `IUIMAConfigurator` service and unsubscribe through the `removeComponentListener()` method of the `IUIMAConfigurator` service.

Methods:

- `componentChanged()` – this method is called by the UIMA Configurator service to notify subscribed applications of the given UIMA component event.

Class `com.ibm.uima.osgi.service.event.UIMAComponentEvent`.

This class encapsulates the attributes of a given UIMA component event. UIMA component event objects are delivered to subscribed listeners when a change occurs in one of the UIMA component bundle's lifecycle. The type code identifies the event type.

Type codes:

- `DISCOVERED` – indicates that the component bundle file is discovered in the designated directory;
- `ERROR` – indicates that there is an error associated with the component bundle;

- **MODIFIED** – indicates that the properties of the component bundle has been modified;
- **REGISTERED** – indicates that the component service is registered in the OSGi Service Registry;
- **REMOVED** – indicates that the component bundle file is removed from the designated directory;
- **UNREGISTERING** - indicates that the component service is being unregistered from the OSGi Service Registry.
- *Methods:*
- `getComponentBundleFile()` – returns the discovered component bundle file, if the event type is **DISCOVERED**; otherwise returns `null`;
- `getComponentBundleManifest()` – returns the `Manifest` object of the removed component bundle file, if the event type is **REMOVED**; otherwise returns `null`;
- `getComponentError()` – returns the component error, if the event type is **ERROR**; otherwise returns `null`;
- `getComponentReference()` – returns the reference to the UIMA component service, if the event type is **REGISTERED**, **MODIFIED** or **UNREGISTERING**; otherwise returns `null`;
- `getType()` – returns the event type code.

Configurator properties file.

As we mentioned earlier, the Configurator module discovers UIMA component bundle files in the designated directory and notify subscribed applications of new UIMA component bundles. To perform this function, the Configurator runs *Directory Scanner* – a thread that periodically scans the designated directory for UIMA component bundle files. The Directory Scanner is configured by using special Configurator properties file. The `uima.configurator.properties` file includes the following configuration properties:

- `uima.osgi.componentDirectory` – the designated directory, which is periodically scanned by the Directory Scanner;
- `uima.osgi.componentDirectoryPollingInterval` - polling interval (ms) for the Directory Scanner.

The default Configurator properties file – `uima.configurator.properties` – is included in the `com.ibm.uima.osgi.service` package. This properties file contains the following parameters:

```
uima.osgi.componentDirectory = uima.home/component_bundles
uima.osgi.componentDirectoryPollingInterval = 10000
```

The 'uima.home' expression here is substituted with the actual path to UIMA home directory, as specified in the following JVM option:

```
-Duima.home=<local_UIMA_home_path>
```

If the default Configurator properties file is used, and the 'uima.home' JVM option is specified, the Directory Scanner will try to scan the `component_bundles` subdirectory of the UIMA home directory for new UIMA component bundles.

Applications can specify a custom Configurator properties file by using the following JVM option:

```
-Duima.configurator.properties=<custom_configuration_file>
```

The properties specified in the `uima.configurator.properties` file can be overridden by the JVM options of the same name.

3.3 UIMA Component Services

Interface `com.ibm.uima.osgi.service.IBundleHook`.

This interface defines general method that allows getting the bundle's internal class loader.

Methods:

- `getBundleClassLoader()` - returns reference to the internal Java class loader of the bundle.

Interface `com.ibm.uima.osgi.service.IUIMAComponent`.

This is the interface of the UIMA component reference service, registered as OSGi service by each active UIMA component bundle.

Methods:

- `getComponentName()` - returns the symbolic name of the component bundle;
- `getComponentSpec()` - returns reference to the component specification service associated with the given UIMA component descriptor path;
- `getComponentSpecs()` - returns references to all the component specification services;
- `getComponentUID()` - returns the UID object identifying this component bundle;
- `getComponentVersion()` - returns the component bundle version;

- `getDefaultComponentSpec()` – returns the reference to the default component specification service; the default component specification service corresponds to the 1st UIMA component descriptor, specified in the component bundle manifest;
- `getProperties()` – returns the full set of the component reference service properties.

Interface `com.ibm.uima.osgi.service.IUIMAComponent.UID`

This interface defines unique ID of the UIMA component bundle.

Methods:

- `getName()` – returns the component bundle symbolic name;
- `getVersion()` – returns the component bundle version;
- `matches()` – returns `true`, if this component UID matches the given component UID.

Interface `com.ibm.uima.osgi.service.IUIMAComponentSpec`

This is the interface of the UIMA component specification service, registered as OSGi service by each active UIMA component bundle.

Methods:

- `getComponentDescriptor()` – returns the URL of the UIMA component descriptor file in the component bundle;
- `getComponentReference()` – returns the reference to the enclosing component bundle's reference service;
- `getComponentSpecUID()` – returns the UID object, identifying this component specification;
- `getProperties()` – returns the full set of the component specification service properties.

Interface `com.ibm.uima.osgi.service.IUIMAComponentSpec.UID`

This interface defines unique ID of the component specification.

Methods:

- `getComponentUID()` – returns the UID of the encapsulating component bundle;
- `getDescriptorPath()` – returns the UIMA descriptor path for the given component specification;
- `matches()` – returns `true`, if this component specification UID matches the given component specification UID.

Interface `com.ibm.uima.osgi.service.IUIMAComponentPublisher`.

This is the interface of the UIMA Component Publisher, registered as OSGi service by the `com.ibm.uima.osgi.service.internal` bundle.

Methods:

- `publishComponent ()` - creates and registers component services for the given UIMA component bundle.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Deutschland Informationssysteme GmbH
Department 0790
Pascalstrasse 100
70569 Stuttgart
Germany

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided

by IBM under terms of the IBM Customer Agreement or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States, other countries, or both.

IBM

The following terms are trademarks or registered trademarks of other companies:

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Eclipse is a trademark of Eclipse Foundation, Inc.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.