

Oracle® OLAP

Developing Analytic Workspace Manager Plug-ins

10g Release 2 (10.2.0.3)

B31205-01

June 2006

This technical note describes how to create a Java plug-in for use with Oracle Analytic Workspace Manager, Version 10.2.0.3.0. With a plug-in, a Java developer can extend the functionality of Analytic Workspace Manager (AWM).

This technical note includes the following topics:

- [Introducing Analytic Workspace Manager Plug-ins](#)
- [AWM Plug-in Interface and Example](#)

See Also:

- *Oracle OLAP Application Developer's Guide*
- *Oracle OLAP Analytic Workspace Java API Reference*
- *Oracle OLAP Developer's Guide to the OLAP API*
- *Oracle OLAP Java API Reference*

Introducing Analytic Workspace Manager Plug-ins

An AWM plug-in enables you to run Java code in the context of AWM. With a plug-in, you can implement programs that perform actions such as the following:

- Create new type of calculations
- Create forecasts
- Implement security measures

In an AWM plug-in, you can use the following Java APIs:

- Oracle OLAP Analytic Workspace Java API
- Oracle OLAP Java API
- Oracle Business Intelligence Beans
- JDBC API
- Swing API

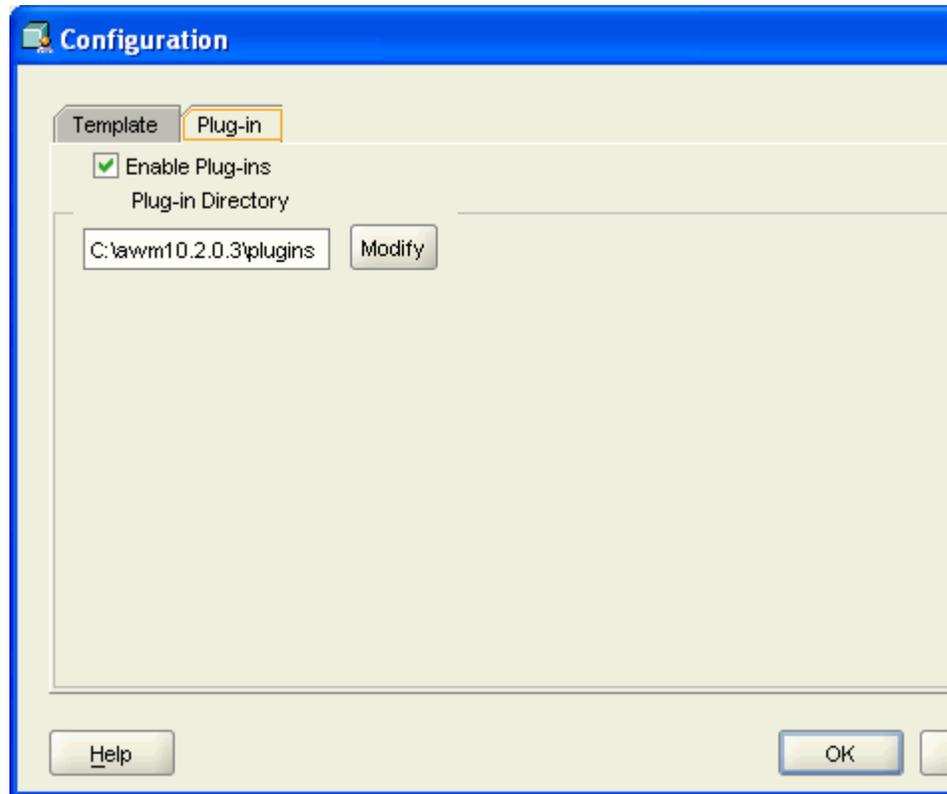
You can invoke OLAP DML or SQL procedures by using JDBC classes.

Enabling AWM Plug-ins

AWM has a configuration option that enables it to use plug-ins. To enable plug-ins, in the AWM Model View, from the **Tools** menu select **Configuration**. On the **Plug-in** tab, select **Enable Plug-ins**. On that tab, the **Plug-in Directory**

field indicates the directory in which AWM looks for plug-ins. [Figure 1](#) shows the **Plug-in** tab of the **Configuration** dialog box with plug-ins enabled.

Figure 1 *Plug-in Tab in Configuration Dialog Box*



How AWM Calls a Plug-in

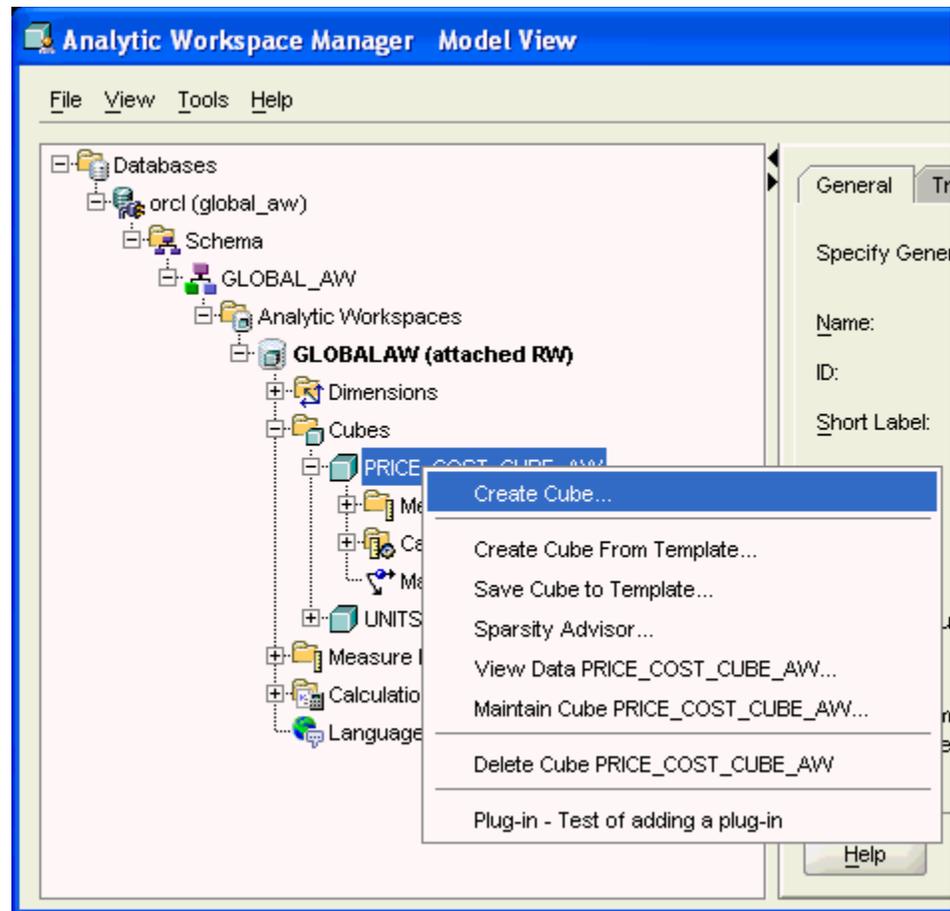
If AWM has plug-ins enabled, then on startup AWM dynamically loads Java code from JAR files located in the plug-ins directory. After loading the contents of the JAR files, AWM looks for classes that implement the `AWMPlugin` interface.

Note: You can include multiple plug-ins in a single JAR file.

When a user right-clicks an item in the AWM Model view navigation tree, a menu appears that presents the actions available for the item. The menu also displays the plug-ins that apply to the item. The plug-in uses the `isSupported` method to indicate whether it applies an item in the tree.

The menu displays the text `Plug-in` - followed by the string returned by the `getMenu` method of the plug-in. [Figure 2](#) shows the menu that AWM displays for a right-click on a Cube in the tree. The menu includes the `PluginTest` example plug-in. For the code of the plug-in example, see [Example 2](#).

Figure 2 Right-click Menu of the AWM Model View Navigation Tree



If the user selects the plug-in, then AWM calls the `handle` method of the plug-in. The `handle` method specifies the actions that the plug-in performs. The `refreshTree` method of the plug-in indicates whether AWM refreshes the navigation tree to include any new items created by the plug-in.

AWM passes the following objects as input parameters to the plug-in methods:

- A `java.sql.Connection` object that represents the current connection to the Oracle Database instance.
- A `java.lang.String` object that indicates the type of the selected AWM navigation tree item. Currently for tree items immediately under the **Schema** item, AWM passes a `String` that contains `SCHEMA`. For all other items, AWM passes `null`.
- A `java.lang.Object` that AWM associates with the item selected in the AWM navigation tree. The `Object` can be a `String`, an object from the `oracle.AWXML` package of the Oracle OLAP Analytic Workspace Java API, or `null`. The `oracle.AWXML` object can be one of the following:

Attribute
Cube
DerivedMeasure
Dimension
Hierarchy

Level
Measure
OlapMeasure
SolveGroup

- An `oracle.AWXML.AW` object that represents the current analytic workspace.
- A `java.util.Map` object that is for use by AWM.

For the `handle` method, AWM also passes a `java.awt.Frame` object that the plug-in can use as the parent frame for user interface components.

AWM does not pass any user identification or password to the plug-in. It only passes the connection object. An AWM plug-in does not allow you to do anything that you cannot do by writing a standalone Java Program.

Steps in Creating a Plug-in

The prerequisites for creating an AWM plug-in are the following:

- Include the `AWMPlugin` interface class in your development environment. You can download a JAR file that contains the `AWMPlugin.class` file from the OLAP OTN Web page <http://www.oracle.com/technology/bi/olap/olap.html>. You could also compile your own `AWMPlugin.class` file by creating the directory structure `oracle/olap/awm/plugin` in your development environment and creating the `AWMPlugin.java` interface in that directory.
- Compile the code with JDK 1.4.

Note: Only plug-ins compiled with JDK 1.4 are compatible with Analytic Workspace Manager in 10g Release 2 (10.2.0.3). A plug-in compiled with JDK 1.5 is not compatible with AWM in this release.

To create an AWM plug-in, do the following:

1. Create a class that implements the `AWMPlugin` interface. In the `handle` method, include the code for the operations that the plug-in performs.
2. Using JDK 1.4, compile the plug-in and any other classes that it uses.
3. Deploy the plug-in and other classes to a JAR file. You can include more than one plug-in in the same JAR file.
4. Put the JAR file in the plug-ins directory.
5. Start AWM.

Note: Each time you put a new or updated version of a JAR file in the plug-ins directory, you must restart AWM. AWM only loads the contents of the JAR files upon startup.

AWM Plug-in Interface and Example

This section contains the specification for the `AWMPlugin` interface and an example that implements the interface.

Plug-in Interface Specification

[Example 1](#) has the code for the interface.

Example 1 The AWMPlugin Interface

```
package oracle.olap.awm.plugin;

import oracle.AWXML.AW;
import java.awt.Frame;
import java.sql.Connection;
import java.util.Map;

/**
 * On startup, AWM looks for classes that implement the AWMPlugin interface.
 * When calling the methods specified by the interface, AWM passes in the
 * following objects:
 *
 * <UL>
 * <LI>A Connection object for the current connection to the Oracle
 * Database instance.
 * <LI>A String object that relates to the type of the selected item in
 * the AWM Model View navigation tree or null.
 * Currently for tree items immediately under the Schema item, AWM
 * passes a String that contains SCHEMA. For all other items, AWM
 * passes null.
 * <LI>An Object that is associated with the selected item in the tree
 * or null. The Object is a String or an oracle.AWXML object.
 * <LI>The oracle.AWXML.AW object for the current analytic workspace.
 * <LI>A Map object that is for future use by AWM.
 * </UL>
 *
 * The isSupported method indicates whether the plug-in applies to an
 * item in the AWM navigation tree.
 * When a user right-clicks a tree object that the plug-in supports, AWM
 * adds the string returned by the getMenu method to the menu it displays.
 * If the user selects the plug-in from the menu, then AWM calls the handle
 * method.
 * When calling the handle method, AWM passes a Frame object that the plug-in
 * can use as the parent Frame for user interface elements it displays.
 */
public interface AWMPlugin
{
    /**
     * Indicates whether the plug-in applies to the selected item in the
     * AWM Model View navigation tree.
     *
     * @param conn The Connection object for the current connection to the
     * Oracle Database instance.
     * @param type A String that is associated with the selected tree item
     * or null.
     * @param obj An Object that is associated with the selected tree item
     * or null. The Object is a String or an oracle.AWXML object.
     * @param aw The AW object for the current analytic workspace.
     */
}
```

```

    * @param params A Map for future use by AWM.
    *
    * @return A boolean that indicates whether AWM should display the
    *         plug-in for the selected item in the tree.
    */
boolean isSupported(Connection conn, String type, Object obj, AW aw,
                    Map param);

/**
 * Gets the String to display on the right-click menu for the plug-in.
 *
 * @param conn The Connection object for the current connection to the
 *             Oracle Database instance.
 * @param type A String that is associated with the selected tree item
 *             or null.
 * @param obj An Object that is associated with the selected tree item
 *            or null. The Object is a String or an oracle.AWXML object.
 * @param aw The AW object for the current analytic workspace.
 * @param params A Map for future use by AWM.
 *
 * @return A String the contains the text that AWM displays on the
 *         right-click menu.
 */
String getMenu(Connection conn, String type, Object obj, AW aw, Map param);

/**
 * Specifies the actions to take when the user selects the plug-in from
 * the navigation tree right-click menu.
 *
 * @param parent A Frame to use as the parent for user interface objects.
 * @param conn The Connection object for the current connection to the
 *             Oracle Database instance.
 * @param type A String that is associated with the selected tree item
 *             or null.
 * @param obj An Object that is associated with the selected tree item
 *            or null. The Object is a String or an oracle.AWXML object.
 * @param aw The AW object for the current analytic workspace.
 * @param params A Map for future use by AWM.
 */
void handle(Frame parent, Connection conn, String type, Object obj,
            AW aw, Map param);

/**
 * Indicates whether AWM should refresh the navigation tree.
 * If the plug-in creates new oracle.AWXML objects, then this method
 * should return true.
 *
 * @param conn The Connection object for the current connection to the
 *             Oracle Database instance.
 * @param type A String that is associated with the selected tree item
 *             or null.
 * @param obj An Object that is associated with the selected tree item
 *            or null. The Object is a String or an oracle.AWXML object.
 * @param aw The AW object for the current analytic workspace.
 * @param params A Map for future use by AWM.
 */
boolean refreshTree(Connection conn, String type, Object obj, AW aw,
                    Map param);
}

```

Example of an AWM Plug-in

[Example 2](#) contains the code for the `PluginTest` class, which implements the `AWMPlugin` interface. The plug-in supports `oracle.AWXML.Cube` and `oracle.AWXML.DerivedMeasure` objects, which correspond to the items under the **Cubes** and **Calculated Measures** items, respectively, of the navigation tree. The plug-in displays a string representation and the class of the `oracle.AWXML` object. It also calls the `getAutoSolve` method of the `Cube` or `DerivedMeasure` object and displays a message determined by the return value of that method. The message appears in a dialog box displayed by `JOptionPane` class. For an example of the message that `PluginTest` displays, see [Figure 3](#).

Example 2 The PluginTest Example

```
import oracle.AWXML.AW;
import oracle.AWXML.Cube;
import oracle.AWXML.DerivedMeasure;
import oracle.olap.awm.plugin.AWMPlugin;

import java.awt.Frame;
import java.sql.Connection;
import java.util.Map;
import javax.swing.JOptionPane;

public class PluginTest implements AWMPlugin
{
    public boolean isSupported(Connection conn, String type, Object obj,
                              AW aw, Map params)
    {
        // Support Cube and DerivedMeasure objects.
        if (obj instanceof Cube || obj instanceof DerivedMeasure)
        {
            return true;
        }
        return false;
    }

    public String getMenu(Connection conn, String type, Object obj, AW aw,
                          Map params)
    {
        // Text to display on the right-click menu.
        String menu = "Test of adding a plug-in";
        return menu;
    }

    public void handle(Frame parent, Connection conn, String type, Object obj,
                       AW aw, Map params)
    {
        String msg = "Plug-in invoked for ";
        String response = "";
        if (obj != null)
        {
            msg += "the " + obj.toString() + " object, ";
            msg += "which is an instance of " + obj.getClass().toString() + ".";
        }
    }
}
```

```

if (type != null)
    msg += "\nThe type is " + type + ".";

if (obj instanceof Cube)
{
    Cube cube = (Cube) obj;
    response = cube.getAutoSolve();
    response = evaluateResponse(response);
    msg += "\n" + response;
}
else if (obj instanceof DerivedMeasure)
{
    DerivedMeasure dmeas = (DerivedMeasure) obj;
    response = dmeas.getAutoSolve();
    response = evaluateResponse(response);
    msg += "\n" + response;
}

JOptionPane.showConfirmDialog(null,
                               msg,
                               "Plug-in Test",
                               JOptionPane.CANCEL_OPTION,
                               JOptionPane.PLAIN_MESSAGE);
}

public boolean refreshTree(Connection conn, String type, Object obj,
                           AW aw, Map params)
{
    return true;
}

public String evaluateResponse(String response)
{
    String isYes = "YES";
    String isNo = "NO";
    String isDefault = "DEFAULT";

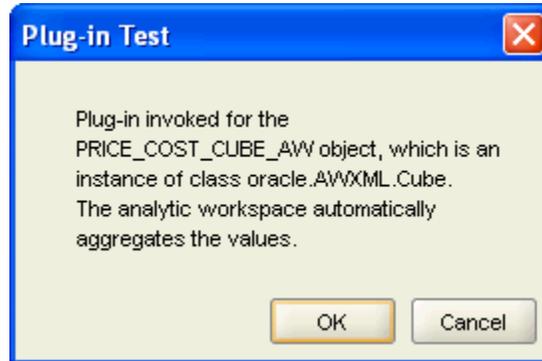
    if(isYes.equalsIgnoreCase(response))
    {
        response = "The analytic workspace automatically aggregates "
            + "the values.";
    }
    else if(isNo.equalsIgnoreCase(response))
    {
        response = "The analytic workspace does not automatically aggregate "
            + "the values.";
    }
    else if(isDefault.equalsIgnoreCase(response))
    {
        response = "Whether the analytic workspace automatically aggregates "
            + "the values of the DerivedMeasure depends on the "
            + "default specified by the parent Cube.";
    }

    return response;
}
}

```

Figure 3 illustrates the type of dialog box that the `PluginTest` example displays.

Figure 3 Dialog Box Displayed by the PluginTest Example



Developing Analytic Workspace Manager Plug-ins, 10g Release 2 (10.2.0.3)
B31205-01

Copyright © 2006, Oracle. All rights reserved.

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

