Oracle Utilities Application Framework

API Reference Guide

Release 25.10

G39827-01

October 2025

ORACLE

Oracle Utilities Application Framework 25.10 API Reference Guide

Contents

Chapter 1

Oracle Utilities Application Framework API Reference Guide

Welcome to the Oracle Utilities Application Framework API Reference Guide. This document describes recommended Oracle Utilities Application Framework (OUAF) Application Programming Interface (API) guidelines and any notable changes, deprecations, and bug fixes between versions.

The APIs described in this document can be used with the following applications:

- Oracle Utilities Customer Care and Billing
- Oracle Utilities Customer to Meter
- Oracle Utilities Digital Asset Management
- Oracle Utilities Meter Data Management
- Oracle Utilities Market Settlements Management
- Oracle Utilities Operational Device Management
- Oracle Utilities Smart Grid Gateway
- Oracle Utilities Work and Asset Management

This document includes the following:

Chapter 2

Introduction

This chapter provides an introduction to Application Programming Interfaces (APIs) provided with the Oracle Utilities Application Framework (OUAF). This includes:

- What are the Public APIs?
- What are *Not* APIs?

What are the Public APIs?

The following packages are designated as public APIs. This code is sanctioned to be called by application customization layers, and consequently, these methods and classes will not likely change across releases. This sanctioned the packages include the name "api" and "domain". In the Oracle Utilities Application Framework, sanctioned packages include:

- com.splwg.base.api.*
- com.splwg.base.domain.*

The same pattern can be applied to the application layer code. For example, in the case of Customer Care & Billing, the package name would change from "base" to "ccb".

What are *Not* APIs?

Avoid calling other packages not mentioned above. The code is not designed to be called outside of OUAF code. For example, avoid the following packages:

- com.splwg.base.support.*
- com.splwg.base.web.*
- com.splwg.base.xai.*

These are often specialized calls intended to be used within the Application Framework. Calling specialized methods may, for example, change processing behavior by introducing unexpected events or skipping optimizations.

Chapter 3

Changes and Deprecations

This chapter outlines notable changes and deprecations in the Oracle Utilities Application Framework (OUAF) Application Programming Interfaces (APIs) and may include changes to public APIs, non-public APIs, and changes caused by other 3rd party libraries or programs. This includes:

OUAF Release 4.0.0.0

com.splwg.base.domain.common.algorithm.Algorithm. getActiveVersion

New Method:

```
com.splwg.base.domain.common.algorithm.AlgorithmComponentCache.get
AlgorithmInfo().getActiveVersionInfo
```

Change Description: This call was changed to improve performance. This new method returns an instance of `com.splwg.base.api.businessObject.AlgorithmInfo`, which can return the active version's information. Alternatively, `com.splwg.base.domain.common.algorithm.AlgorithmComponentCache` can directly be used to instantiate any `AlgorithmComponent` implementation.

com.splwg.base.support.algorithm.AbstractAlgorithmComponent. setAlgorithm

Change Description: This method now requires a `com.splwg.base.api.businessObject.AlgorithmInfo`, a new object introduced to improve performance by caching the Algorithm information.

com.splwg.base.support.algorithm.AbstractAlgorithmComponent. setParametersFrom

Change Description: This method now requires a `com.splwg.base.api.businessObject.AlgorithmInfo`, a new object introduced to improve performance by caching the Algorithm information.

OUAF Release 4.0.2.0

com.splwg.base.support.FrameworkSession.clear

Change Description: This new method requires a string to describe the reason the clear method was called. It was added to aid in diagnosing performance issues.

Note: This is a non-public API.

com.splwg.base.support.FrameworkSession.flush

Change Description: This new method requires a string to describe the reason the clear method was called. It was added to aid in diagnosing performance issues.

Note: This is a non-public API.

com.splwg.base.domain.common.maintenanceObject. MaintenanceObjectInfomationAlgorithmSpot.setEntity

Change Description: This method was changed from `setEntity(BusinessEntity<?> entity)` to `setEntity(BusinessEntity entity)`. The change was needed to support Java 6 (and may possibly have been a JDK 6 bug).

java.util.HashSet Behavior Change

Change Description: A behavior change in Java 6 exposed an issue that may require code changes. The iteration order of the HashSet changed. While HashSet does not guarantee any iteration order, there have been instances of code that relied on the old behavior.

In Hibernate, a given table/column can only participate in one property mapping. If a table/column is included in more than one property mapping (based on the defined constraints), the artifact generator must determine which property mapping to use. In prior versions of the artifact generator, based on Java version 5, if two constraints went to the same table and used some of the same fields, the preference order was determined

based on the order that the constraints were returned from a database query. In OUAF 4.x, which is based on Java 6, that order comes out differently, resulting in a different property mapping that broke the product upgrade. This is a rare situation, but whenever it arises, the property mapping must come out the same as before the upgrade to avoid breaking the upgraded product.

A check was added to the artifact generator to always deterministically identify the property mapping. If the artifact generator cannot make a definitive decision, this results in the following error:

Constraints should differ in at least some way.

To address the error, the developer should decide which constraint should be followed and which should be "ignored".

Choose a field that is in the constraint to be ignored that is not in the other constraint, and add it to the annotation for the entity implementation class. For example,

```
* @BusinessEntity (cobolPersistence = true, tableName = CI_RC,
ignoredFKFields =
{"STEP_RC_SEQ"},
```

OUAF Release 4.1.0.0

Deprecation: com.splwg.base.api.batch.AbstractThreadWorker.rollbackToSave Point

New Method: Use the `com.splwg.base.api.SavepointExecutable` class (see next item, `AbstractThreadWorker.setSavePoint`).

Deprecation: com.splwg.base.api.batch.AbstractThreadWorker.setSavePoint

New Method: Use the `com.splwg.base.api.SavepointExecutable` class.

Code Example:

```
//Before Code: doSomething1(); doSomething2(); try() {
setSavepoint("BeforeUpdate"); doSomething3();
} catch (Exception e) { rollbackToSavepoint("BeforeUpdate");
doSomething4();
}

//After code: doSomething1(); doSomething2();
SavepointExecutable executable = new SavepointExecutable() {
@Override
protected void execute() { doSomething3();
}
};
SavepointResult result = executable.doIt(savepointName); if
(result.hasError()) {
doSomething4();
}
```

Note: This method does not support proper nesting of savepoints.

OUAF Release 4.2.0.0

com.splwg.base.api.batch.AbstractThreadWorker.getActiveBatch Thread

New Method:
`com.splwg.base.api.batch.AbstractThreadWorker.getBatchThreadMBean`

Change Description: The method name was changed to `getBatchThreadMBean`. This was a name change only.

com.splwg.base.support.context.FrameworkSession. getConnection

Change Description: The upgrade of hibernate changed the connection handling semantics wherein the unit of work is passed in to hibernate rather than obtaining a connection.

Code Example 1 (preferred):

```
//Before Code:
  FrameworkSession session =
  (FrameworkSession)SessionHolder.getSession();Connection
  connection = FrameworkSession.getConnection(session);

  // work with the
  connection
  connection.prepareStatemen
  t();
  …

//After code:
   PreparedStatement query = createPreparedStatement(sql...);
```

Code Example 2 (uses internal APIs):

```
//Before Code:
  FrameworkSession session =
  (FrameworkSession)SessionHolder.getSession();Connection
  connection = FrameworkSession.getConnection(session);

  // work with the
  connection
  connection.prepareStatemen
  t();
  …

//After code:
  ((FrameworkSession) getSession()).doWork(new ConnectionWork() {

      @Override
      public void execute(Connection connection) throws SQLException {

          // work with the
          connection
          connection.
          prepareStatement()
```

```
                ;
                ...
            }
        });
```

OUAF Release 4.2.0.1

Bug Fix: com.splwg.base.support.context.HibernateCacheHelper. flushAndClear

Bug: The OUAF 2.2 method was not implemented in the initial OUAF Release 4.0.0.0. It has since been added with patch 16580225.

> **Note**: This is a specialized call intended for a specific case in which a COBOL program is needed to clear the hibernate cache because it could not issue commits. It is not recommended for general use. Patch 7447321 contains information about this method.

Bug Fix: com.splwg.base.api.PreparedStatement (Constructor)

Bug: This method was removed in OUAF Release 4.2.0.1 and was restored with patch 17793307.

OUAF Release 4.2.0.2

Bug Fix: com.splwg.base.domain.common.masterConfiguration. MasterConfiguration_Id (Constructor)

Bug: The method signature was changed in OUAF Release 4.2.0.2 and was restored with patch 18347676.

Bug Fix: com.splwg.base.api.PreparedStatement (Constructor)

Bug: This method was removed in OUAF Release 4.2.0.1 and was restored with patch 17802274.

Bug Fix: com.splwg.base.domain.common.businessObject. MessageRepository.businessObjectStatusOptionReplacedBy

Bug: This method was removed in OUAF Release 4.2.0.1 and was restored with patch 18277216.

OUAF Release 4.3.0.0.0

Bug: This change was made to pass down the action details to `BusinessObjectEnterStatusAlgorithmSpot` and `BusinessObjectExitStatusAlgorithmSpot`.

17230588 - EXPOSE ACTION IN BO STATUS PLUG-IN SPOTS

com.splwg.base.domain.common.businessObject. BusinessObjectEnterStatusAlgorithmSpot

New Method:

```
/**
   * Set the Action of Business Object
   */
 void setAction(BusinessObjectActionLookup boAction);
```

Change Description: The method is added to the plugin spot to pass down the action information to all implementation classes.

> **Note**: As the spot is changed, all implementation classes should implement the method. Implementation can be left blank if the algorithm does not need the action information.

Code Example 1 (preferred):

```
//Before Code:
   There was no setAction() method.

//After code:
  public void setAction(BusinessObjectActionLookup boAction) {
 // null
   }
```

com.splwg.base.domain.common.businessObject. BusinessObjectExitStatusAlgorithmSpot

New Method:

```
/**
   * Set the Action of Business Object
*/
 void setAction(BusinessObjectActionLookup boAction);
```

Change Description: The method is added to the plugin spot to pass down the action information to all implementation classes.

> **Note**: As the spot is changed, all implementation classes should implement the method. The implementation can be left blank if the algorithm does not need the action information.

Code Example 1 (preferred):

```
//Before Code:
   There was no setAction() method.

//After code:
```

```
 public void setAction(BusinessObjectActionLookup boAction) {
// null
  }
```

Bug Fix: JS API - handleExpand/handleCollapse Methods in xslInclude.js

Bug: `handleExpand()` and `handleCollapse()` functions in xslInclude.js used to expand/collapse a zone. Prior to Framework 4.3, these functions expected one or two arguments to expand/collapse a zone. However, since the release of Framework 4.3, these functions do not work as expected with a single argument. They now accept only two arguments, e.g., sequence id of the zone, and user event object, to work as expected. The API change was a part of overall standardization and adoption of best practices for the JS layer to enable cross-browser support. Refer to patch 23745289 for details.

OUAF Release 4.3.0.2.0

Removed Two-decimal Money Limitation

Bug: A change was made to allow for 0, 1, 3, or more decimal places for money objects. Code that is incorrectly comparing number values using the Java "equals()," method must be changed to use the `compareTo()` or `isEqualTo()` methods. This code might have worked in the past due to the hard-coding of two decimal places. A money value of "2" would be returned as "2.00". This is no longer true, since the decimal places for money are now based on metadata, and returns the proper number of digits.

21788123 - INVOICING - CURRENCY / DECIMAL ISSUES

Incorrect Code Example:

```
assertEquals(algComp.getTotalAmount(), new Money(adjustmentAmount,
newCurrency_Id("USD")));
```

Correct Code:

```
assertEquals(algComp.getTotalAmount().compareTo(new
Money(adjustmentAmount, newCurrency_Id("USD"))), 0);
```

OUAF Release 4.3.0.4.0

API to access Oracle JET

Bug: A new API to access Oracle JET (OJET) in OUAF-based applications has been added as a part of a bug fix. The fix abstracts OJET libraries from direct use in edge products and CM code. The changes leverage existing OUAF mechanisms to simplify, consolidate, abstract, and optimize the usage of OJET in OUAF applications. These changes help minimize the impact of OJET changes and complexities in OUAF-based applications.

25507178 - OPTIMIZE OJET INCLUSION

Include the new UI map fragment F1-OJETLIBS in the HTML and invoke one of the methods (such as `init()`) of the new JavaScript class, **`ouaf.ui.OJETHelper`**.

HTML Example:

```
<oraInclude map="F1-OJETLIBS"/>
…

<script type=”text/javascript”>
ouaf.ui.OJETHelper.init(null, false, ['ojs/ojknockout',
     'ojs/ojchart'],function ($, oj, ko) {
   …

     }
);
   </script>
```

Also, to access JQuery, please use the $ symbol directly in the code. JQuery is loaded by OUAF in all applications and is available throughout the application as '$'. It is recommended that you use the JQuery instance provided by OUAF and not include any JQuery libraries explicitly.

OUAF Release 24.1.0.0.0

Enhanced Composite ID Class Generation to Output Optional ID Constructors

Oracle Utilities Application Framework 24.1 implemented a new BusinessEntity annotation -> `generateOptionalIdConstructors=true`

which will work with any nullableKeyFields to generate extra public ID Constructors which will only require the non nullable key fields as input and default the nullable fields to null.

Note this flag will only make sense if there exists some nullableKeyFields defined.

With both of these annotations defined the Code generation for an ID class will change as follows

1. A new static collection optionalFields will be generated containing a list of strings representing the nullable keys.
2. New public ID constructor(s) will be generated with only the non nullable keys as parameters which internally will set each nullable key field to null
3. Within the existing public constructor which accepts a Map<String, Object> valuePairs parameter, call a new method to augment the valuePairs with any missing nullable key entries setting them to null prior to calling the super(valuePairs) constructor method.
4. Add new static method optionallyAddDefaultKeys(...) to add any missing nullable key entries setting them to null

This will enable any existing references to new MasterConfiguration_Id(BusinessObject) or new MasterConfiguration_Id(BusinessObjectId)

To continue to work after the new CONFIG_PART_NAME column is added as an optional primary key to the table/entity.

Also changes to the table java classed need to handle the style of nullable key entities when retrieving FK References.

These changes are

1. When constructing construct a collection of the optionalKey fields and a count of the number of required key fields
2. When performing the validation "Too few PK Values for table " only check if the number of keys provided satisfys the required key count
3. Prior to attempting to construct the ID, If there are some optional fields and the key collection is less that the full key size then pad the collection out with blank entries