

Framework for Package API-Based ADF Business Components

User's Guide

The Framework for Package API-Based ADF Business Components allows you to declaratively create entity and view object definitions which use a package API, rather than standard SELECT and DML statements, to perform queries from and updates of the database. While it should be useful in and of itself, its primary purpose is to provide a demonstration of how, using custom declarative properties, Java code in Oracle ADF Business Components can be generalized and factored up to a framework of classes.

Contents

Importing the Framework into a Project.....	1
Using the Framework Classes as Base Classes for Your Project.....	2
Setting Up Entity Object Definitions that Use Package APIs Instead of DML Operations	3
Setting Up View Object Definitions that Use Package APIs Instead of SELECT Statements.....	8
Setting Up View Link Definitions for Package API-Based View Object Definitions.....	11
Setting Up Associations for Package API-Based Entity Object Definitions	12
GNU Free Documentation License.....	13

Importing the Framework into a Project

Before you can use the framework in your own application, you must import it into the Model project of your application as a library. This procedure assumes you have downloaded the framework JAR file (package-api-framework.jar) and the framework Javadoc file (package-api-framework-doc.jar) to your file system.

1. In the Application Navigator, double-click your Model project node to open the Project Properties dialog.
2. Select the Libraries and Classpath page.
3. Click **Add Library** to open the Add Library dialog.
4. Click **New** to open the Create Library dialog.
5. Select **Deployed by Default** unless you intend to place the library directly on your Java EE container's classpath.

6. Select the Class Path node and click **Add Entry**.
7. Browse to the location of package-api-framework.jar, select it, and click **Select**.
8. Select the Doc Path node and click **Add Entry**.
9. Browse to the location of package-api-framework-doc.jar, select it, and click **Select**.
10. Optional: If you have downloaded the source code for the framework, you can enable browsing/tracing into the framework source by doing the following:
 - a. Select the Source Path node and click **Add Entry**.
 - b. Browse to the location of the downloaded source code.
 - c. Open the Framework/src directory and click **Select**.
11. Click **OK** until all dialogs are closed.
12. Repeat this entire procedure for your ViewController project. Alternatively, you can choose to share the library you created for the Model project and add that same library directly to the ViewController project.

Using the Framework Classes as Base Classes for Your Project

The procedure for using the framework classes as base classes for your project varies depending on whether you are already using custom framework classes that extend the ADF base classes. If you do not know whether you are using custom framework classes, you probably aren't.

If You Are Already Using Custom Framework Subclasses

This procedure applies if you are using custom framework classes for any of the following:

- Entity object [row] class
- Entity object definition class
- View object class
- View object definition class

For each of the above custom framework classes you are using:

1. Open the source code for the class in an editor.
2. Change the class so that, rather than extending a class in the oracle.jbo.server package, it extends one of the following:
 - com.quovera.packageapi.EntityImpl (for entity object classes)
 - com.quovera.packageapi.EntityDefImpl (for entity definition classes)
 - com.quovera.packageapi.ViewObjectImpl (for view object classes)
 - com.quovera.packageapi.ViewDefImpl (view view definition classes)

If You Are Not Already Using All of the Above Custom Framework Subclasses

This procedure applies if you are not using all of the above custom framework subclasses:

1. Double-click your Model project to open the Project Properties dialog.
2. Select the Business Components\Base Classes page.

3. Under Entity Object, fill the following fields with the corresponding values, unless you are already using a custom framework class for that field:
 - **Row** as "com.quovera.packageapi.EntityImpl"
 - **Definition** as "com.quovera.packageapi.EntityDefImpl"
4. Under View Object, fill the following fields with the corresponding values, unless you are already using a custom framework class for that field:
 - **Object** as "com.quovera.packageapi.ViewObjectImpl"
 - **Definition** as "com.quovera.packageapi.ViewDefImpl"
5. Click **OK**.

NOTE: If you have already created entity or view object definitions in your project before registering the framework classes, you may need to "touch" each entity/view object definition in the project (by opening it in an editor and then saving) to force it to use the classes. Entity and view object definitions created after the base classes are registered will automatically use the framework classes.

Setting Up Entity Object Definitions that Use Package APIs Instead of DML Operations

If you want to create an entity object definition that uses Package APIs instead of DML operations, you can do one of two things:

- Base the entity object definition on a database object (table, view, or synonym), just as you would most entity object definitions. This is an ideal option if such a database object exists, and is necessary if you only want to use package APIs for some (rather than all) database writes.
- Create an entity object definition as you would for forward generation of a database table. Your entity object definition should, at the least, contain persistent attributes for any parameter that needs to be passed to package APIs in lieu of calling INSERT, UPDATE, or DELETE statements. The table name for the entity object definition does not matter in this case; the framework will ignore it.

Database Package API Requirements for Entity Object Definitions

Exactly which package APIs you must specify for an entity object definition depends on whether or not the entity object definition is based on a database object and, if it is, which functions you want to perform using the package API.

Insert, Update, and/or Delete Package Procedures

If you want to use a package API in lieu of INSERT, UPDATE, and/or DELETE operations for your entity object instances, you must provide appropriate package procedures. These procedures should accept some or all attributes from the entity object instance as IN or INOUT parameters. Any entity object attributes that must be refreshed after insert or update (such as sequence-based primary key attributes) should be present as OUT or INOUT parameters. If your entity object is not based on an underlying database object, or is based on a database

view without appropriate INSTEAD OF triggers, you must provide all three package procedures unless you intend to block insertion, change, or update of rows at the application level.

In the sample application, the package EMPLOYEES_PKG provides the following procedures, which are used to in lieu of INSERT, UPDATE, and DELETE operations for the Employees entity object instance:

```
PROCEDURE CREATE_EMP (P_EMPLOYEE_ID OUT NUMBER,  
                     P_NAME VARCHAR2,  
                     P_EMAIL VARCHAR2,  
                     P_JOB_ID VARCHAR2,  
                     P_DEPARTMENT_NAME VARCHAR2) ;  
  
PROCEDURE UPDATE_EMP (P_EMPLOYEE_ID NUMBER,  
                     P_NAME VARCHAR2,  
                     P_EMAIL VARCHAR2,  
                     P_JOB_ID VARCHAR2,  
                     P_DEPARTMENT_NAME VARCHAR2) ;  
  
PROCEDURE DELETE_EMP (P_EMPLOYEE_ID NUMBER) ;
```

Select Package Function

If your entity object definition is not based on a database object, you will need to define a special “select” package function for your entity object definition, if either of the following applies:

- You are using a package procedure in lieu of UPDATE operations, and you have cross-attribute validation logic that may trigger attribute fault-in (where attributes not originally queried by a view object instance are requested and must be retrieved from the database).
- You are using a package procedure in lieu of UPDATE and/or DELETE operations, and you are using any locking mode except LOCK_NONE, including pessimistic or optimistic locking. If you don't know your locking mode, you are probably using pessimistic or optimistic locking.

The select package function can accept any number of IN parameters, which will be filled by attribute values from a particular entity object instance. The function should return a REF CURSOR containing a single row; this row should contain values for each entity object attribute. That is, by passing in a primary key or other unique combination of values, the application must be able to retrieve a complete set of values from the row.

In the sample application, the package EMPLOYEES_PKG provides the GET_EMP_BY_PK function, which is used to select data for a particular Employees entity object instance:

```
TYPE MY_REF IS REF CURSOR;  
FUNCTION GET_EMP_BY_PK (P_EMPLOYEE_ID NUMBER) RETURN MY_REF;
```


Lock Package Procedure

If you are using a package procedure in lieu of UPDATE and/or DELETE operations, and you are using any locking mode except LOCK_NONE, including pessimistic or optimistic locking, you need to provide a lock procedure. This procedure should accept some or all attributes from the entity object instance as IN parameters, and should perform whatever operations are needed to acquire a lock on underlying database data.

In the sample application, the package EMPLOYEES_PKG provides the following procedure, which is used to lock data for a particular Employees entity object instance:

```
PROCEDURE LOCK_EMP (P_EMPLOYEE_ID NUMBER) ;
```

Setting Up an Entity Object Definition to Use an Insert Procedure

To set up an entity object definition to use a package procedure to create rows, follow these steps:

1. Open the entity object definition in an editor.
2. On the General page, click the Add (+) icon for the "Custom Properties" section and select "Non-translatable Property" from the dropdown list to add a new line to the table.
3. Enter the *Property* as "**InsertProc**" (without quotes).
4. Enter the *Value* as the package-qualified name of your insert procedure; e.g., "employees_pkg.create_emp" (without quotes).
5. Open the Attributes page of the editor.
6. Select the attribute that corresponds to the first parameter in the insert procedure.
7. Click the Add (+) icon for the "Custom Properties: AttributeName" section.
8. Enter the *Property* as "**InsertIndex**" (without quotes).
9. Enter the *Value* as "1" (without quotes).
10. Repeat steps 6-9 for each attribute that corresponds to a parameter in the insert procedure, entering *Value* as the 1-based parameter index ("2" for the attribute corresponding to the 2nd parameter and so on).

The following steps apply only if the entity object definition contains attributes with Refresh On Insert set (including all attributes of type DBSequence):

11. If any attributes with Refresh On Insert correspond to OUT only parameters (rather than INOUT parameters) for the insert procedure, select the first of them.
12. Click the Add (+) icon for the "Custom Properties: AttributeName" section.
13. Enter the *Property* as "**OutOnlyInsert**" (without quotes). You can use any value for the *Value* property, but the property must be present.
14. Repeat steps 11-13 for any other attributes corresponding to OUT only parameters.

Setting Up an Entity Object Definition to Use an Update Procedure

To set up an entity object definition to use a package procedure to update rows, follow these steps:

1. Open the entity object definition in an editor.
2. On the General page, click the Add (+) icon for the "Custom Properties" section and select "Non-translatable Property" from the dropdown list to add a new line to the table.
3. Enter the *Property* as "**UpdateProc**" (without quotes).
4. Enter the *Value* as the package-qualified name of your update procedure; e.g., "employees_pkg.update_emp" (without quotes).
5. Open the Attributes page of the editor.
6. Select the attribute that corresponds to the first parameter in the update procedure.
7. Click the Add (+) icon for the "Custom Properties: AttributeName" section.
8. Enter the *Property* as "**UpdateIndex**" (without quotes).
9. Enter the *Value* as "1" (without quotes).
10. Repeat steps 6-9 for each attribute that corresponds to a parameter in the update procedure, entering *Value* as the 1-based parameter index ("2" for the attribute corresponding to the 2nd parameter and so on).

The following steps apply only if the entity object definition contains attributes with Refresh On Update set:

11. If any attributes with Refresh On Update correspond to OUT only parameters (rather than INOUT parameters) for the update procedure, select the first of them.
12. Click the Add (+) icon for the "Custom Properties: AttributeName" section.
13. Enter the *Property* as "**OutOnlyUpdate**" (without quotes). You can use any value for the *Value* property, but the property must be present.
14. Repeat steps 11-13 for any other attributes corresponding to OUT only parameters.

Setting Up an Entity Object Definition to Use a Delete Procedure

To set up an entity object definition to use a package procedure to delete rows, follow these steps:

1. Open the entity object definition in an editor.
2. On the General page, click the Add (+) icon for the "Custom Properties" section and select "Non-translatable Property" from the dropdown list to add a new line to the table.
3. Enter the *Property* as "**DeleteProc**" (without quotes).
4. Enter the *Value* as the package-qualified name of your update procedure; e.g., "employees_pkg.delete_emp" (without quotes).
5. Open the Attributes page of the editor.
6. Select the attribute that corresponds to the first parameter in the delete procedure.
7. Click the Add (+) icon for the "Custom Properties: AttributeName" section.
8. Enter the *Property* as "**DeleteIndex**" (without quotes).
9. Enter the *Value* as "1" (without quotes).

10. Repeat steps 6-9 for each attribute that corresponds to a parameter in the delete procedure, entering *Value* as the 1-based parameter index ("2" for the attribute corresponding to the 2nd parameter and so on).

Setting Up an Entity Object Definition for Package API-Based Select and Lock

If you have supplied select and/or lock APIs for your entity object definition (as described in the section, "[Database Package API Requirements for Entity Object Definitions](#)," you must configure your entity object definition to use them. Follow these steps:

1. Open the entity object definition in an editor.
2. On the General page, click the Add (+) icon for the "Custom Properties" section and select "Non-translatable Property" from the dropdown list to add a new line to the table.
3. Enter the *Property* as "**SelectFn**" (without quotes).
4. Enter the *Value* as the package-qualified name of your select function; e.g., "employees_pkg.get_emp_by_pk."
5. If you are using a lock procedure, repeat steps 2-4 to add a Property called "**LockProc**" with a value of the package-qualified name of your lock procedure; e.g., "employees_pkg.lock_emp".
6. Open the Attributes page of the editor.
7. Select the attribute that corresponds to the first parameter in the select function.
8. Click the Add (+) icon for the "Custom Properties: AttributeName" section.
9. Enter the *Property* as "**SelectIndex**" (without quotes).
10. Enter the *Value* as "1" (without quotes).
11. Repeat steps 7-10 for each attribute that corresponds to a parameter in the select function, entering *Value* as the 1-based parameter index ("2" for the attribute corresponding to the 2nd parameter and so on).
12. Repeat steps 7-11 for the lock procedure, using the property name "**LockIndex**".
13. Select the attribute that corresponds to the first *column* in the REF CURSOR returned by the select function.
14. Click the Add (+) icon for the "Custom Properties: <AttributeName>" section.
15. Enter the *Property* as "**TargetIndex**" (without quotes).
16. Enter the *Value* as "1" (without quotes).
17. Repeat steps 14-16 for each attribute that corresponds to a column in the REF CURSOR.

Setting Up View Object Definitions that Use Package APIs Instead of SELECT Statements

You can create a SQL-only or entity-based view object definition, based on standard or package-based entity object definitions, and configure it to use a package function instead of issuing its SELECT statement. If you do this, any query specified as part of the view object definition will be ignored.

Database Package API Requirements for View Object Definitions

Only one database package function—to perform the query—is required for package-based view object definitions, but a separate function to return a count of rows the query is expected to return is strongly recommended.

Select Package Function

The purpose of a package-based view object definition is to use a package function to supply the data collection for the view object. This package function should contain only IN parameters (strictly speaking, it can contain INOUT parameters, but the OUT values of parameters will be ignored), and should return a REF CURSOR containing functions that correspond to any non-transient attributes of the view object. The REF CURSOR need not contain these columns in order, and it may contain extra columns (beyond what the view object will use).

If the view object definition is to parametrize its query through named bind variables or view link instances, the underlying package function must accept appropriate parameters. For example, the EMPLOYEES_PKG package used by the demo application provides the function

```
FUNCTION GET_EMPS (P_DEPT_NAME VARCHAR2, P_JOB_ID VARCHAR2) RETURN  
MY_REF;
```

This function allows for querying employees by department name, job ID, both, or neither. If the function did not accept parameters, it would not be possible to restrict its rowset using named bind variables or view links.

Count Package Function

ADF needs to be able to retrieve the number of database rows that will be processed into view rows. In the case of a standard SQL-based view object instance, it calculates this number using a COUNT(*) query.

We strongly recommend providing a package function that can perform an equivalent task for package-based view object instances. This function can take parameters (in general, it should take the same parameters as does the query function, although this is not required) and should return a NUMBER that provides an accurate count of the number of rows that will be returned if the select package function is executed.

If you do not provide such a function, the framework will continue to work correctly, but will execute the select function, load all its rows into middle-tier memory, and count them whenever a row count is needed. This will significantly degrade performance, especially for select functions returning large numbers of rows.

In the sample application, the package EMPLOYEES_PKG provides the following function, which is used to provide a row count for the GET_EMPS function:

```
FUNCTION COUNT_EMPS (P_DEPT_NAME VARCHAR2, P_JOB_ID VARCHAR2) RETURN  
NUMBER;
```

Setting Up a View Object Definition to Use a Select Function

To set up a view object definition to use a package function to select rows, follow these steps:

1. Open the view object definition in an editor.
2. On the General page, click the Add (+) icon for the "Custom Properties" section and select "Non-translatable Property" from the dropdown list to add a new line to the table.
3. Enter the *Property* as "**SelectFn**" (without quotes).
4. Enter the *Value* as the package-qualified name of your select function; e.g., "employees_pkg.get_emps."
5. Open the Query page of the editor.
6. Click the Add (+) icon for the "Bind Variables" section to open the Bind Variable dialog.
7. On the Variable page of the dialog, add a bind variable (with appropriate Java type) corresponding to the first parameter to the select function.
8. Open the Custom Properties page of the dialog.
9. Enter the *Property* as "**SelectIndex**" (without quotes).
10. Enter the *Value* as "1" (without quotes).
11. Click **Add** to add the custom property.
12. Click **OK**.
13. Repeat steps 6-12 for each parameter to the select function, entering *Value* as the 1-based parameter index ("2" for the 2nd parameter and so on).
14. Open the Attributes page in the editor.
15. Select the attribute that corresponds to the first *column* in the REF CURSOR returned by the select function.
16. Click the Add (+) icon for the "Custom Properties: <AttributeName>" section.
17. Enter the *Property* as "**TargetIndex**" (without quotes).
18. Enter the *Value* as "1" (without quotes).
19. Repeat steps 15-18 for each attribute that corresponds to a column in the REF CURSOR.

Setting Up a View Object Definition to Use a Count Function

If you are using a function to select rows, we strongly recommend using a count function as well. To do so, follow these steps:

1. Open the view object definition in an editor.
2. On the General page, click the Add (+) icon for the "Custom Properties" section and select "Non-translatable Property" from the dropdown list to add a new line to the table.
3. Enter the *Property* as "**CountFn**" (without quotes).
4. Enter the *Value* as the package-qualified name of your select function; e.g., "employees_pkg.get_emps."
5. Open the Query page of the editor.
6. If your count function uses any parameters not used by your select function, create bind variables for those parameters (see steps 6-7 of [the preceding procedure](#)).
7. Select the bind Variable corresponding to the first parameter of the count function and click the Edit (pencil) icon to open the Bind Variable dialog.
8. Open the Custom Properties page of the dialog.

9. Enter the *Property* as **CountIndex** (without quotes).
10. Enter the *Value* as "1" (without quotes).
11. Click **Add** to add the custom property.
12. Click **OK**.
13. Repeat steps 7-11 for each parameter to the count function, entering *Value* as the 1-based parameter index ("2" for the 2nd parameter and so on).

Setting Up View Link Definitions for Package API-Based View Object Definitions

If you want to use a package API-based view object definition on an end of a view link definition for which there is no accessor (for example, for the source of a standard unidirectional view link definition), you need not do anything special. Simply create the association or view link definition as normal. However, if you want to use a package-based association on an end of a view link definition for which there is an accessor (for example, for the destination of a standard unidirectional view link definition, or for either end of a bidirectional view link definition), you must configure the view object definition appropriately.

Database Package API Requirements for Linking View Object Definitions

As explained previously, if the view object definition is to parameterize its query through view link instances, the select package function must accept appropriate parameters. This includes at least one parameter corresponding to every passed-in value from the view link's other end.

For example, in the demo application, the view link definition `EmployeesJobsLink` is used to access employees by job ID; the job ID value to query is passed in from the other end of the link. Because of this, the select function for `EmployeesView` accepts (in addition to a separate parameter) a parameter that corresponds to the job ID to query (if NULL is passed in, the function returns employees regardless of job).

Configuring a View Object Definition to Be Accessed via a View Link

To set up a view object definition which uses a package API and participates in a view link definition as a side for which there is an accessor, complete the following steps:

1. Open the view object definition in an editor.
2. Open the Query page.
3. Select the bind variable corresponding to the first value that will be passed through the view link (in the demo, for `EmployeesJobsLink`, this is `PJobId`) and click the Edit (pencil) icon to open the Bind Variable dialog.
4. Enter the *Property* as **VLParam**.
5. Enter the *Value* as "Bind_<MasterAttrName>", where <MasterAttrName> is the name of the attribute from the view object definition at the *other* end of the view link definition which provides the relevant value (for the `PJobId` variable, this attribute [from `JobsView`] is `JobId`, so the value is `Bind_JobId`).
6. Click **Add** to add the custom property.
7. Click **OK**.

8. Repeat steps 3-7 for any other bind variables corresponding values that will be passed through the view link.

Setting Up Associations for Package API-Based Entity Object Definitions

For many cases, you may not need to do anything special to create associations involving package API-based entity object definitions. You will only need to specifically set up associations if *both* of the following are true:

- The association contains an accessor that retrieves instances of a package API-based entity object definition (default associations are bidirectional, so both entity object definitions have instances retrieved by accessors; in a standard unidirectional association, only the destination entity object definition has instances retrieved by accessors).
- The (or one of the) package API-based entity object definitions are *not* based on database views.

In this case, you must set up business components accordingly.

Creating View Object Definitions to Use in the Association

If you have an entity object definition that requires special setup for the association, you *must* create a view object definition based on that entity object definition. In most cases, the view object definition will itself require a package API; see the section, "[Setting Up View Object Definitions that Use Package APIs Instead of SELECT Statements](#)" for information about how requirements for the API and how to set up a view object definition to use it.

In particular, the select function must accept, among its parameters, a parameter for every value that will be passed through the association.

If you have a view object definition that already satisfies these requirements, you do not need to create another such definition specifically for association use.

NOTE: You will experience some improvement in performance if you use or create a view object definition that exposes all those attributes in the entity object definition that you will need to access for most rows.

Configuring an Association to Use the Custom View Object Definition

Once you have a view object definition for the entity object definition requiring special setup, you can configure the association to use that view object definition with the following steps:

1. Open the association in an editor, and open the Tuning page.
2. Click the Edit (pencil) icon to open the Custom Views editor.
3. Select "Use Custom View Object" for the entity object definition requiring special setup (or for both entity object definitions, if both require special setup).

4. Select a view object definition satisfying the requirements discussed earlier in this section.
5. Click OK.

Configuring a View Object Definition for Use in the Association

To set up a view object definition which uses a package API and is used in an association as above:

1. Open the view link definition in an editor.
2. Open the Query page.
3. Select the bind variable corresponding to the first value that will be passed through the association and click the Edit (pencil) icon to open the Bind Variable dialog.
4. Enter the *Property* as "VLParam".
5. Enter the *Value* as "Bind_<MasterAttrName>", where <MasterAttrName> is the name of the attribute from the entity object definition at the *other* end of the association which provides the relevant value.
6. Click **Add** to add the custom property.
7. Click **OK**.
8. Repeat steps 3-7 for any other bind variables corresponding values that will be passed through the association.

GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <<http://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

Package-Based ADF BC Framework User's Guide

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all

Package-Based ADF BC Framework User's Guide

these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

Package-Based ADF BC Framework User's Guide

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

Package-Based ADF BC Framework User's Guide

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

Package-Based ADF BC Framework User's Guide

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C) YEAR YOUR NAME.
```

```
Permission is granted to copy, distribute and/or modify this document
```

```
under the terms of the GNU Free Documentation License, Version 1.3
```

```
or any later version published by the Free Software Foundation;
```

```
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
```

```
A copy of the license is included in the section entitled "GNU
```

```
Free Documentation License".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with ... Texts." line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
```

```
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.