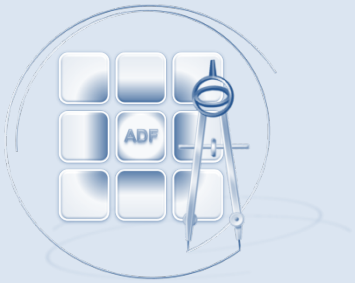
ORACLE

ADF Architecture Square

ADF Naming and Project Layout Guidelines v2.00 6/Feb/2013



twitter.com/adfArchSquare

Abstract

This document sets out ADF object naming conventions and project layout guidelines for ADF applications built with ADF Business Components and ADF Faces Rich Client (release 11g and greater). The conventions have the goal of giving developers and development teams a short circuit on producing their own collateral.

The document is not a final production, but a living document that will be extended to cover new information as discovered or the ADF framework changes.

Readers are encouraged to discuss this document on the ADF EMG (http://bit.ly/JOr1wW) and provide constructive feedback to Chris Muir via the ADF EMG Issue Tracker (http://bit.ly/Qj2JAw).

Author: Chris Muir
Date: 6/Feb/2013

Introduction

This document sets out ADF object naming conventions and project layout guidelines for ADF applications built with ADF Business Components and ADF Faces Rich Client (release 11g and greater). The conventions have the goal of giving developers and development teams a short circuit on producing their own collateral if it doesn't already exist.

The document is not a final production, but a living document that will be extended to cover new information as discovered or the ADF framework changes.

Oracle encourages you to use this document as a set of guidelines that you extend from for your own projects, note which guidelines are relevant to your projects and those that are not, and if you find the document unsatisfactory for whatever reason share this back with the original authors and the wider ADF community via the ADF EMG issue tracker http://bit.ly/Qj2JAw.

Readers should be aware in using this document that many of the guidelines are obvious because they are what JDeveloper generates by default. Yet other guidelines are defined purely arbitrarily by the author (arbitrarily: Determined by chance, whim, or impulse, and not by necessity, reason, or principle). As example for View Objects stored in a Model project this document dictates they go under a package suffix ".views". You may prefer ".queries". There is no right or wrong answer, just a need for readers to understand that this document has to define some guideline, and as a result one is supplied.

Generally the guidelines should be followed but in the cases where it doesn't make sense to do so, where you have your own preference, certainly diverge from the guidelines. However ensure to document why and when this has occurred so your team follows a single guideline rather than an ambiguous set of conflicting guidelines from Oracle and your own efforts.

In terms of the "currency" of the guidelines in this document, note this document was written and published when JDeveloper 11.1.1.6.0 and 11.1.2.3.0 were the latest available releases of JDeveloper in the 11gR1 and 11gR2 branches. Obviously in the future new features may be included in later releases that may make certain guidelines obsolete or invalid. Always attempt to seek the latest version of this document, and, if you detect a situation where the document is invalid please attempt to contact the original authors to share your concerns via the ADF EMG issue tracker http://bit.ly/Qj2JAw.

Goals

"Guidelines defined: Noun: A general rule, principle, or piece of advice."

The goals of this document are:

- Ease development and maintenance of ADF projects
- Make ADF code easier to read and understand by providing a frame of reference to the ADF community
- Be flexible enough to adapt to change
- Applicable to multiple JDeveloper versions
- Easy to refer to
- Extensible

Principles

This document follows 2 broad IT conventions in order to create a consistent set of naming guidelines:

- Convention over Configuration – also known as Coding by Convention (as defined by Wikipedia) is a software design paradigm which seeks to decrease the number of decisions that developers need to make, gaining simplicity, but not necessarily losing flexibility.

  To some readers the fact an Oracle and ADF document is trying to dictate convention over configuration might seem rather comical given ADF through its heavy use of XML files is all configuration. However the leading principle of decreasing the number of decisions developers need to make is easily adaptable into this guide.

- No Hungarian Notation – Hungarian notation (as defined by Wikipedia) is a code convention of adding prefixes or suffixes to objects to indicate the object types or use. For example for an integer counter field it would be named iCounter. While it's recognized some programmers & languages have a preference for using Hungarian notation, because it is arguable not the norm in Java programming, and it can lead to inconsistent names when code is modified or ported, it will not be adopted for this document as it is counter to the stated goal "Be flexible enough to adapt to change".

  There are some scenarios where the principles conflict with each other and it is necessary to apply a precedence order to the rules.

  For example when JDeveloper creates a View Object, it includes a “View” suffix in the View Object’s name. This violates the No Hungarian Notation principle. But to change the name violates the Convention over Configuration principle.

  To resolve this conflict, the document puts precedence on Convention over Configuration before the No Hungarian Notation rule.

  This has the added benefit that when you introduce junior programmers to your ADF team who have just undertaken ADF training, or a new ADF programmer joins your team from another organization, they will be familiar with the default conventions used by JDeveloper over a unique set of guidelines that require you to modify all names.

Document Structure

This document is structured by major topic areas, starting with general naming guidelines, workspace and project guidelines, packages and directory structures, followed by specific guidelines for ADF Business Component and ViewController projects separately. Each area is broken down into an alphabetical list of minor topics for each major topic area.

Broadly speaking the document attempts to not repeat itself, moving repeated topics into the general guidelines area at the beginning of the document. Feel free to leap in and read any topic in this document, but realize that your reading needs to also consider the broader general guidelines which may also have information to the relevant topics at hand.

Future Extensions

This document is designed to be a living document that will be revised and updated as new guidelines come to light, as well as new subject areas are introduced and covered.

Document Annotations

If you choose to adopt the guidelines from this document each has been annotated so you may refer back to them from your own documentation and development code.

Each guideline has been given a unique identifier that follows a specific syntax:

ADFng<major-guideline-version>-<identifier>

- ADFng - Stands for "Application Development Framework Naming Guidelines", essentially this guide.
- ADFng number prefix - This document uses a major-point-minor version numbering scheme such as v1.00 or v4.03. The number before the decimal point is the major version number, the number after the minor version number within the major version.

  Major revisions to this document such as the introduction of guidelines will result in a new major version number. Minor revisions to this document such as the correction of spelling, grammar or similar will result in a minor revision number updates only.

  The number directly after the ADFng annotation refers to which major version number of this document the recommendation was introduced, or, if the recommendation was modified from the last version, the version number the recommendation was last modified. As such [ADFng3-12345] is a guideline either introduced or last modified in v3 of this document.
- The remaining number after the hyphen is a unique number across all versions of this document. As such if you see guideline numbers [ADFng1-12345] and [ADFng1-12346] these are two distinct guidelines, however if you see numbers [ADFng1-12345] and [ADFng3-12345] these two guidelines are one and the same, the later number being a revision of the 12345 guideline at version 3.

  Over time it is expected there will be multiple releases of this document with new recommendations added, existing recommendations modified and obsolete recommendations removed. The following conventions will be followed for each guideline:
- Each guideline annotation (that is [ADFng1-12345]) will be unique across all releases of the documentation. A guideline annotation will never be reused.
- If a guideline's text is significantly modified (beyond spelling mistakes and grammar fixes) between versions of this document the ADFng number prefix will be updated to the current major version of this document. For example the recommendation ADFng1-06000 introduced in release 1 of this document and updated in release 6 of this document will become ADFng6-06000. If the recommendation is not subsequently updated in release 7, the recommendation annotation will remain at ADFcg6-06000.
- If a guideline becomes obsolete, the text will be removed except for the recommendation annotation marked as obsolete.

  Beyond the annotations used above, for some guidelines you'll note the guideline has a footnote that refers to an external document. Rather than duplicating the complete text from the external document, it's left up to the reader to pursue and read the links.

Change Log

The following changes have been made in this v2.00 of the guidelines:

Rule	Change	ADF EMG Issue #
ADFng2-02008	Modified rule to include naming convention for different deployment profile needs/use.	ADFEMG-88
ADFng2-02009	Modified rule to include naming convention for different deployment profile needs/use.	ADFEMG-88
ADFng2-02010	Fixed rule such that deployment profile names are in lowercase as per JEE standard, and a clarification required to remove the need/use from the artifact name as relating to rules 02008 and 02009.	ADFEMG-102 ADFEMG-88
ADFng2-03005	Fixed rule such that the default ADF BC model package is com.acme.hr.*model*.*adfbc*	ADFEMG-89
Post rule ADFng1-03016	This change hasn't resulted in a change to any rule as it applies to a paragraph explaining some general options for all rules ADFng1-03006 to 03016. The explanation describes where the rules 03006-03016 can be configured, and has not changed the definition of the rules.	ADFEMG-90
ADFng2-03017	Removed incorrect assumption public_html is not configurable and a Java EE standard. Documented configuration option.	ADFEMG-86
ADFng2-03021	Added that pages/fragments of a BTF should be placed under public_html/WEB-INF so they can't be accidentally accessed directly bypassing the ADF controller	ADFEMG-87
ADFng2-03025	Reworded slightly to imply unbounded task flow files as which there can be many, separate to the unbounded task flow which at runtime there is only one	ADFEMG-92
ADFng2-03026	Same change as ADFng2-03021 above.	ADFEMG-87
Post rule ADFng1-03027	This change hasn't resulted in a change to rule ADFng1-03027, but rather a clarification following the table the rule is contained in to explain how the rule can be implemented.	ADFEMG-93
ADFng2-03029	New rule to define location of web.xml javax.servlet.http.HttpSessionListener or java.servlet.Filter extensions in ViewController project	

ADFng2-04025	Clarified rule to imply the Java class of the managed bean as it was ambiguous	ADFEMG-96
ADFng2-04027	Minor typo fixed	-
ADFng2-04029	This rule was overloaded with more than one rule and as such has been split into this rule and rule ADFng2-04064. See the note for that rule in this table	
ADFng1-04036	This rule has been deleted as it is deemed to provide no benefit, is not essential, and requires a lot of work by developers	ADFEMG-91
ADFng2-04059	Reworded slightly to imply unbounded task flow files as which there can be many, separate to the unbounded task flow which at runtime there is only one	ADFEMG-92
ADFng2-04064	This is a new rule split from rule ADFng1-04029 which was overloaded with two rules.	

The ADF EMG Issues can be seen here: http://java.net/jira/browse/ADFEMG

General Naming Guidelines

The following naming guidelines provide the general rules to apply when coding with ADF. Following this section are guidelines which detail rules for specific ADF object types.

- [ADFng1-01000] - **All guidelines successfully followed** - this document is full of negatives, don't do this, don't do that. It's nice to start with a positive don't you think? If you comply with all the guidelines in this document you can apply this "success" code.
- [ADFng1-01001] – **Keep names comprehensible** – throughout this document there are obviously numerous naming guidelines. In the following guidelines it's important to not follow them blindly if the suggested naming convention will create incomprehensible names.

  For example many databases use meaningless three letter abbreviation for representing each table in a foreign key which can be hard for programmers to discern. When creating an ADF BC Entity Object Association the rule in this document is to replicate the foreign key name with the suffix "Assoc". However there's no point replicating the meaningless foreign key names in ADF, instead replace it with something developers can discern.
- ADF objects should attempt to follow the Java SE naming conventions as published by Oracle Corporation[1] (reproduced here in its entirety):

Rule	Identifier Type	Rules for Naming	Examples
[ADFng1-01002]	Packages	The prefix of a unique package name is always written in all-lowercase ASCII letters and should be one of the top-level domain names, currently com, edu, gov, mil, net, org, or one of the English two-letter codes identifying countries as specified in ISO Standard 3166, 1981. Subsequent components of the package name vary according to an organization's own internal naming conventions. Such conventions might specify that certain directory name components be division, department, project, machine, or login names.	com.sun.eng com.apple.quicktime.v2 edu.cmu.cs.bovik.cheese
[ADFng1-	Classes	Class names should be nouns, in	class Raster;

[1] Code Conventions for the Java TM Programming Language - Revised April 20, 1999 - Section 9 Naming Conventions - http://bit.ly/javasecodeconvsect9

01003]		camelcase with the first letter of each internal word capitalized. Try to keep your class names simple and descriptive. Use whole words-avoid acronyms and abbreviations (unless the abbreviation is much more widely used than the long form, such as URL or HTML).	class ImageSprite;
[ADFng1-01004]	Interfaces	Interface names should be capitalized like class names.	interface RasterDelegate; interface Storing;

[ADFng1-01005]	Methods	Methods should be verbs, in camelcase with the first letter lowercase, with the first letter of each internal word capitalized.	run();
			
			runFast();
			
			getBackground();

[ADFng1-01006]	Variables	Except for variables, all instance, class, and class constants are in camelcase with a lowercase first letter. Internal words start with capital letters. Variable names should not start with underscore _ or dollar sign $ characters, even though both are allowed by the compiler. Variable names should be short yet meaningful. The choice of a variable name should be mnemonic- that is, designed to indicate to the casual observer the intent of its use. One-character variable names should be avoided except for temporary "throwaway" variables. Common names for temporary variables are i, j, k, m, and n for integers; c, d, and e for characters.	int i; char c; float myWidth;

[ADFng1-01007]	Constants	The names of variables declared class constants and of ANSI constants should be all uppercase with words separated by underscores ("_"). (ANSI constants should be avoided, for ease of debugging.)	static final int MIN_WIDTH = 4;
			
			static final int MAX_WIDTH = 999;
			static final int GET_THE_CPU = 1;

- [ADFng1-01008] – **Non- Java object names** - The guideline for Classes should extend to non-Java specific objects you create in JDeveloper too.

  For example JDeveloper workspaces should also use camelcase with the first letter capped, such as StudentEnrolmentSystem

- [ADFng1-01009] - **Acronyms in object names** - If a name contains an acronym, only capitalize the first letter of the acronym rather than all the letters.

  For example "RegisteredTfn" for Registered Tax File Number rather than "RegisteredTFN".

- [ADFng1-01010] - **Meaningful object names** - Object names should be meaningful even when they're taken out of context.

  For example naming pages like A122.jspx, C400.jspx means little and only a close inspection of the application and code would show their use. Alternatively names like EnrolStudent.jspx and CancelReservation.jspx convey more information on what they're used for without looking at the application.

- [ADFng1-01011] – **Common abbreviations in object names** - Use abbreviations where the abbreviation is more commonly used than the long form of a name

  For example use HTML and URL over HyperText Markup Language and Uniform Resource Locator.

- [ADFng1-01012] - **Unnecessary abbreviation in object names** - Avoid unnecessary abbreviation to assist readability unless the object name is large.

  For example DateOfBirth is easier to read than Dob. However use common sense. For example ExtensionToSeries700LegislationErrata would be clumsy to work with, so ExtSeries700Legislation may be better in this case.

- [ADFng1-01013] - **Object name consistency** - If you use acronyms or abbreviations, be consistent.

  For example don't refer to Tfn in one location and TaxFileNumber in another

- [ADFng1-01014] - **Don't use Hungarian notation** - Hungarian notation[2] is a code convention of adding prefixes or suffixes to objects to indicate the object types or use. For example for an integer counter field it would be named iCounter. While it's recognized some programmers & languages have a preference for using Hungarian notation, because it is arguable not the norm in Java programming, and it can lead to inconsistent names when code is modified or ported, it will not be adopted for this document as it is counter to the stated goal "Be flexible enough to adapt to change".

  The exception to this is where JDeveloper by default uses such notations by default. As example on creating view objects typically JDeveloper would include the suffix "View". In this case such conventions will be maintained as they are familiar to all ADF programmers reducing the learning curve, and to change from this convention requires more work than necessary.

[2] Wikipedia Hungarian Notation - http://en.wikipedia.org/wiki/Hungarian_notation

There are situations where the database schema you've based your ADF BC project uses a Hungarian notation of its own, such as naming tables with a "T_" prefix like "T_CUSTOMERS". In this case this document gives you conflicting rules as it tells you to not use Hungarian notation, but at the same time your Entity Objects should be the same name as your table name in initcapped camel-case (resulting in TCustomers for the previous example). Which rule to follow is up to you as you need to assess which one makes more sense for your situation. For example if you discovered only half your tables used the T_ prefix, then maybe now is a good time to drop the prefix in ADF.

Workspace and Project Guidelines

The following sections of this document consider the naming guidelines for the IDE artifacts specific to Oracle JDeveloper, namely workspaces, project and packages.

Workspaces

Within context of Oracle JDeveloper, "Workspaces" are also known as "Application Workspaces" or more generically the "Application". When naming your workspaces you should consider:

- [ADFng1-02000] - **Workspace and directory names** - The workspace name should be the same name and case as the base directory for the application on the file system and the relating .jws file.

  For example a workspace named VehicleLicensingSystem should have a correlating VehicleLicensingSystem.jws file located with the directory VehicleLicensingSystem.

- [ADFng1-02001] - **Meaningful workspace names** - The application workspace name should reflect the application name known by the users.

  For exampe VehicleLicensingSystem or Procurement.

- [ADFng1-02002] - **Unique workspace names** - The application workspace name should be unique across all application workspaces at your organization.

Guidelines for Workspaces Types

Oracle ADF provides a flexible solution for defining the architecture for your application. A recommended presentation on the topic can be found here: http://bit.ly/adfarchsqangels

This document does not recommend any set architecture but attempts to provide naming guidelines to follow regardless of your architecture. Depending on your ADF application architecture your application may contain one workspace for the application, or at the other extreme hundreds of workspaces, one containing common ADF framework reusable components such as page templates, skins, model components, another set of workspaces containing reusable task flows, and then the final consuming application workspace that brings all of these together.

If you choose to implement a single workspace the naming guidelines above are satisfactory. If you implement a larger multi-workspace application, the following guidelines should also be followed:

- [ADFng1-02003] - **Common workspace names** - Excluding workspaces that just contain bounded task flows, if a workspace's contents are reusable elements such as page templates, skins, or ADF BC model components, the workspace name for these should include Common as a prefix.
- [ADFng1-02004] - **Skin workspace names** - Depending on how you architect your application, you may place a hierarchy of skins in one workspace, or split them out into separate workspaces. If you do the later consider naming the base skin workspace with the name Base<name>Skin so it is obvious which skin is the top of the hierarchy.

Projects

Oracle JDeveloper workspaces are divided into projects which contain the source code of your applications. The following naming guidelines apply:

- [ADFng1-02005] - **Project and directory names** - Project names should be the same name and case as the base directory for the project on the file system and the relating .jpr file.

  For example the ViewController project should have a collaborating ViewController.jpr file stored in a ViewController directory on the file system under your workspace directory.

- [ADFng1-02006] - **Unique project names per workspace** - Ensure project names are unique per workspace.

Database Connections

Naming guidelines that apply to Database Connections names:

[ADFng1-02007] - **Database Connection names** - Each database connection defined within an application or the IDE name start with the name of the database schema they connect to and end with the suffix "Conn".

Deployment Profiles

JDeveloper allows at both the application workspace level and project level to define deployment profiles which are used in producing different Java archive such as JAR, WAR and EAR files. The following naming guidelines apply to the deployment profile themselves and the archives they produce:

- [ADFng2-02008] – **Application level deployment profile names** – For deployment profiles created at the application workspace level, the deployment profile name should be the same as the application workspace name. If you have more than one deployment profile that correspondence to different needs/use, include the need/use as a suffix such that it takes the form <WorkspaceName><Need>.

  For example: Procurement or ProcurementTest or ProcurementProd

- [ADFng2-02009] – **Project level deployment profile names** – For deployment profiles created at the project level, the deployment profile name should take the form <WorkspaceName><ProjectName. If you have more than one deployment profile that correspondence to different needs/use, include the need/use as a suffix such that it takes the form <WorkspaceName><ProjectName><Need>.

  For Example: CommonModel or CommonModelTest or EmpViewControllerProd.

- [ADFng2-02010] - **Deployment profile archive name and file extensions** - the archive name and file extension that is produced for the relating deployment profile should be the same as the deployment profile name minus the need/use as defined in rules 02009 and 02010, all in lowercase as per the Java convention[3], with a use of an appropriate file type. For example:

  - ADF Library <deploymentprofilename>.jar commonmodel.jar
  - EAR File <deploymentprofilename>.ear procurement.ear
  - JAR File <deploymentprofilename>.jar utilities.jar

[3] Java Guidelines, Patterns, and code for end-to-end Java applications - http://bit.ly/WwRsLQ

- WAR File <deploymentprofilename>.war hr.war

Packages and Directory Structures

Oracle JDeveloper makes use of Java packages in projects for organizing files within the projects. Packages and directory structures are inherently related and as such both are covered here.

Configuring Workspace and Project Packages

As this section dictates packages for workspaces and projects, it's worthwhile explaining how and where the package settings can be configured beforehand.

When creating a JDeveloper workspace via the Create New Application Wizard it will prompt you for the Application Package Prefix where the organization package prefix and default package name for the application can be entered. This is later stored in the Application Properties - Application Content - Package Prefix option and can be modified there.

For JDeveloper projects each is typically set when you create the projects either via a Create Application Wizard or associated Create Project Wizards. Alternatively the default package can be changed via Project Properties – Project Source Path – Default Package option after the project has been created.

Specifically for ADF Business Component projects, the sub packages used by different ADF BC component types are placed in can be configured via Tools Preferences – ADF Business Components – Packages.

Workspace

The following naming guidelines apply to the packages used by the workspace:

- [ADFng1-03000] – **Workspace default package organization name** - As per the Java SE naming conventions a default organization name prefix should be added to all JDeveloper Java code packages and this needs to be decided early on in your development for all ADF application development. For code that will be externalized typically you use your organization's name as a prefix. For example if your commercial organization owned and used the domain name "acme.com", the relating default Java package prefix would be "com.acme.".

  You'll note Oracle with its own packages (e.g. oracle.jbo.server) drops the domain name category from the package prefix (e.g. .com). Most large software vendors do this with the security that it is a trademark. From your own organization's perspective you need to ensure if you drop the domain category, that your code won't conflict with another organizations that has the same name but different domain category (e.g. shopping.com vs shopping.org). If you're not sure, as a safe measure include the category.

- [ADFng1-03001] – **Workspace default package name** - The workspace default package should take the following form:

  <organization.reverse.domain.name>.<workspace-name>

  As a result the following examples show some appropriate package names based on this convention:

  - "Common" workspace com.acme.common
  - "Employees" task flow workspace com.acme.employees
  - "Hr" application workspace com.acme.hr

- [ADFng1-03002] - **Unique workspace package name** - The complete application package name should be unique across all JDeveloper application workspaces in your organization.
- [ADFng1-03003] - **Workspace name package abbreviation** - The previous guideline entitled "Meaningful workspace names" dictated that the application workspace name should reflect the application name known by the users such as VehicleLicensingSystem or Procurement. As the workspace name is part of the default package name for the workspace, using a long and meaningful workspace name in the package can become difficult to use in the IDE. In these cases it may be necessary to use an abbreviated workspace name within the package name.

  For example com.acme.vls for the VehicleLicensingSystem workspace.

ADF Business Component Projects

ADF Business Components projects should make use of the following guidelines:

- [ADFng1-03004] - **Base ADF BC source directory** - Under the ADF BC project all ADF BC packages should be based under the default source directory created by JDeveloper "src".

  The "src" directory is not readily viewable in the Application Navigator. To view this directory select the Navigator Display Options button on the Application Navigator then select Group By Directory. The "src" directory will then appear under the "Application Sources" directory in your Model project within the Application Navigator.

  The "src" directory is configured via the Model Project Properties - Project Source Paths - Java Source Paths option.
- [ADFng2-03005] – **Base ADF BC project package** – In extension to the default package name for the application workspace, the default package prefix for ADF BC model projects should include "model.adfbc". For example if your base application workspace name is "com.acme.common", then the relating default package prefix for your ADF BC model project would be "com.acme.common.model.adfbc".
- ADF Business Component projects should make use of the following package guidelines:

Rule	Object	Package	Example
[ADFng1-03006]	Model.jpx file	Resides in base package for project	com.acme.hr.model.adfbc.Model.jpx
[ADFng1-03007]	Framework extensions	.base	com.acme.hr.model.adfbc.base.HrEntityImpl.java
[ADFng1-03008]	Application modules	.services	com.acme.hr.model.adfbc.services.AppModule
[ADFng1-03009]	Diagrams	.diagrams	com.acme.hr.model.adfbc.diagrams.AdfBcDiagram
[ADFng1-03010]	Domains	.domains	com.acme.hr.model.adfbc.domains.EmailDomain
[ADFng1-	Entity objects	.entities	com.acme.hr.model.adfbc.entities.

03011]			Employees
[ADFng1-03012]	Entity associations	.entities.associations	com.acme.hr.model.adfbc.entities.associations.EmpDeptFkAssoc
[ADFng1-03013]	Property sets	.properties	com.acme.hr.model.adfbc.properties.ReadOnlyPropertySet
[ADFng1-03014]	Validation rules	.validations	com.acme.hr.hrmodel.adfbc.validations.EmailValidationRuleImpl.java
[ADFng1-03015]	View objects	.views	com.acme.hr.hrmodel.adfbc.views.EmployeesView
[ADFng1-03016]	View links	.views.links	com.acme.hr.hrmodel.adfbc.views.links.EmpDeptFkLink

Typically the sub packages can be defined when each ADF BC object is created via the relevant wizard. A permanent configuration can be set via the Tools menu > Preferences > Business Components > Packages.

ViewController Projects

ViewController projects should make use of the following directory guidelines:

- [ADFng2-03017] - **Adhere to base ViewController source directories** - Under an ADF ViewController project JDeveloper creates the followings directories to store source code relating to the ViewController project. Developers should adhere to placing files in the relevant locations:

    1. adfmsrc - The adfmsrc directory is designed to contain all the files associated with the binding layer, specifically adfm.xml, DataBindings.cpx and the relating PageDef.xml files.
    2. src - The src directory is designed to contain all the Java file in your ViewController project, typically your JSF beans.

       While JDeveloper is capable of working with files intermixed between the src and adfmsrc locations, developers should follow the JDeveloper conventions described in the previous paragraph.
    3. public_html - The public_html directory is for source code that relates to rendering UI content including pages, page fragments, page templates, declarative components, skins, task flows, task flow templates, as well as JEE required files including faces-config.xml, trinidad-config.xml and web.xml.

These directories are not readily viewable in the Application Navigator. To view these directory select the Navigator Display Options button on the Application Navigator then select Group By Directory. The directories will then appear under the "Application Sources" directory in your Model project within the Application Navigator.

For reference the "adfmsrc" and "src" directories can be configured via the ViewController Project Properties - Project Source Paths - Java Source Paths option, while "public_html" is configurable via Project Properties - Project Source Paths - Web Application - HTML Root Directory.

- [ADFng1-03018] - **Images, css and js directories** - Place images, css and js files under subdirectories of the same name in the ViewController/public_html directory. Ensure to configure the resource servlet in the web.xml file for these directories[4]:

```
<servletmapping>
  <servletname>resources</servletname>
  <urlpattern>/images/*</urlpattern>
</servletmapping>
<servletmapping>
  <servletname>resources</servletname>
  <urlpattern>/css/*</urlpattern>
</servletmapping>
<servletmapping>
  <servletname>resources</servletname>
  <urlpattern>/js/*</urlpattern>
</servletmapping>
```

- [ADFng1-03019] – **Base ViewController project package** – In extension to the default package name for the application workspace, the default package prefix for ViewController projects should include ".view". For example if your base application workspace name is "com.acme.hr", then the relating default package prefix for your ViewController project would be "com.acme.hr.view".

Bounded Task Flows

The task flows defined within your ViewController project are related to four different artifacts:

1. The task flow XML file that represents the task flow itself
2. The pages or (page) fragments defined by the task flow
3. Indirectly the pageDef XML files of the pages and fragments
4. The Java code/beans used by the task flows, pages and fragments

Each of these is stored in a different area of the ViewController project by default:

1. ViewController/public_html/WEB-INF
2. ViewController/public_html
3. ViewController/adfmsrc

[4]ADF -Real World Performance Tuning by Duncan Mills - http://bit.ly/MillsAdfPerf

4. ViewController/src

As your ViewController project may include more than one task flow the number of source code artifacts within each of these locations becomes significantly larger and potentially harder to comprehend where they relate. As such a naming and directory scheme to organize the artifacts split across the four locations can reduce this issue.

In defining such a scheme it must be mindful of a potential issue in defining reusable bounded task flows. Reusable bounded task flows can be published in ADF Libraries and consumed by other ADF applications. It is a requirement when consuming the reusable bounded task flows that their ID, that is their namespace, is unique. If this convention isn't followed ADF has no way to distinguish between the separate flows.

A bounded task flow is uniquely identified by a combination of the following components:[5]

1. Path to the task flow definition file relative to the HTML /WEB-INF root
2. Name of the task flow definition file
3. ID of the task flow within the definition file

Thus a typical region binding might look something like the following:

```
<taskFlow id="AddressTaskFlow"
          taskFlowId="/WEB-INF/TaskFlows/AddressTaskFlow.xml#AddressTaskFlow">
```

Returning to the directory and naming scheme to relate the separate artifacts of a task flow, and in understanding how task flows are unique identified, the task flow namespace provides a convention for relating the disparate artifacts and is proposed in the following guidelines.

Please note within Oracle Corporation there are different schools of thought on structuring the artifacts, but there is no one-size-fits-all answer, there are multiple solutions each with advantages and disadvantages. The following solution is one that requires no refactoring after the source code has been generated, but does create long directory structures based around the task flow IDs.

The following table summarizes the rules with regards to the directory structures and packages to use for the bounded task flows and their relating objects:

Rule	Object	Package/Directory	Example
[ADFng1-03020]	Bounded task flow XML file	ViewController/public_html/WEB-INF/<default project package>/<boundedtaskflowname>	ViewController/public_html/WEB-INF/com/acme/hr/view/searchdepartmentstaskflow
[ADFng2-	Bounded task flow page	ViewController/public_	ViewController/public_html/com

[5] Task Flow Design Fundamentals - Section Naming Considerations - http://bit.ly/adftaskflowfund

03021]	or fragments	html/WEB-INF/<default project package>/<boundedtaskflowname>	/acme/hr/view/searchdepartmentstaskflow/FindDepartment.jsff
[ADFng1-03022]	pageDef XML files for pages, page fragment or task flow data bounded method calls of the bounded task flow	ViewController/adfmsrc/<default project package>/<boundedtaskflowname>	ViewController/adfmsrc/com/acme/hr/view/searchdepartmentstaskflow/FindDepartmentsPageDef.xml
[ADFng1-03023]	Java classes including beans used by the bounded task flow	ViewController/src/<default project package>/<boundedtaskflowname>	ViewController/src/com/acme/hr/view/searchdepartmentstaskflow/AuditQuery.java

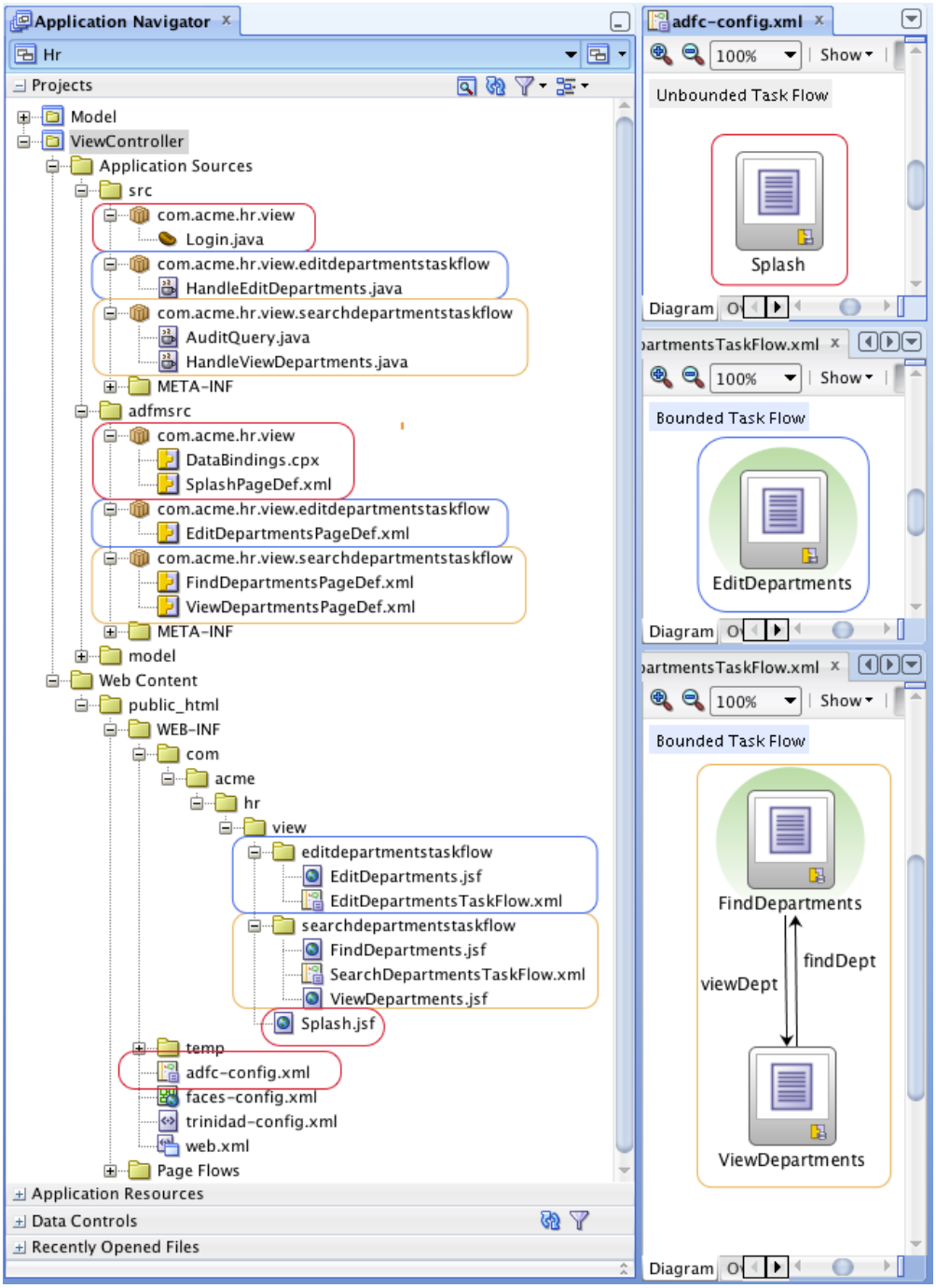
Given the above conventions for bounded task flows, broadly speaking we want the convention for unbounded task flow to be the same. However unbounded task flows differ in one significant fact in that the default adfc-config.xml file (and any other unbounded task flow files but typically an application only has 1) must exist in the root ViewController/public_html/WEB-INF directory exactly. In acknowledging the "rootness" quality of the unbounded task flow we can equate this to the base project directory (e.g. com/acme/hr/view), and for the other artifacts relating to the unbounded task flow we can place them in the base project directory such as "com/acme/hr/view" rather than creating a directory like "com/acme/hr/view/unboundedtaskflow".

Based on this convention the following dictates the rules for the unbounded task flow and its artifacts:

Rule	Object	Package/Directory	Example
[ADFng1-03024]	Default unbounded task flow adfc-config.xml file	ViewController/public_html/WEB-INF	ViewController/public_html/WEB-INF/adfc-config.xml
[ADFng2-03025]	Optional additional unbounded task flow files	ViewController/public_html/WEB-INF	ViewController/public_html/WEB-INF/adfc-config1.xml
[ADFng2-03026]	Unbounded task flow pages	ViewController/public_html/WEB-INF	ViewController/public_html/Splash.jspx
[ADFng1-03027]	pageDef XML files for pages or task flow data bounded method calls of the unbounded task flow	ViewController/adfmsrc/<default project package>	ViewController/adfmsrc/com/acme/hr/view/SplashPageDef.xml
[ADFng1-03028]	Java classes including beans used by the unbounded task flow	ViewController/src/<default project package>	ViewController/src/com/acme/hr/view/Login.java

Note for rule ADFng1-03027 this will require the pageDef files to be moved via their default location using the File menu Rename option.

The following screenshot shows an example of unbounded and bounded task flow artifacts in the IDE. Within the diagram note the colors used to assist visualize the grouping of the task flows to their artifacts:



- ViewController projects should make use of the following package guidelines:

Rule	Object	Package	Example
[ADFng2-03029]	Extension to javax.servlet.http.HttpSessionListener or javax.servlet.Filter configured via web.xml	com.acme.hr.view.servlet	com.acme.hr.view.servlet. MyHttpSessionListener

ADF Business Component Naming Guidelines

The following naming guidelines are recommended for ADF Business Components objects:

Application Modules

Naming guidelines that apply to ADF Business Component application modules:

- [ADFng1-04000] - **Application module name suffix** - Application module names should always include the suffix "AppModule".

  For example HrAppModule.

- [ADFng1-04001] - **Application module view object instances** - Individual instances of view objects exposed through an application module should follow the "convention over configuration" guideline and maintain the same name as the view object name they are based on.

  Each time the view object is used within an application module the instance name is given a numerical suffix by the IDE. These suffixes should be removed. If the same view object is required twice in an application module give the instance a meaningful name that differentiates its use from the sibling view object.

  If a view object instance has a view criteria attached when defined in the application module the view object instance name should reflect the restriction applied by the criteria.

  For example a view object instance that attaches a view criteria CurrentStaffViewCriteria to only return the current staff members from the view could use the name CurrentStaffView.

- [ADFng1-04002] - **Application module view link instances** - Individual instances of view link instances exposed through an application module should follow the "convention over configuration" guideline and maintain the same name as the view link name they are based on.

  For example a View Link entitled EmpDeptFkLink should have a similar name if exposed through the Application Module.

  Each time the view link is used within an application module the instance name is given a numerical suffix by the IDE. These suffixes should be removed. If the same view link is required twice in an application module give the instance a meaningful name that differentiates its use from the sibling view links.

Domains

Naming guidelines that apply to ADF Business Component domains:

- [ADFng1-04003] - **Domain name suffix** - Domains should always have the suffix "Domain".

For example EmailDomain.

Entity Objects

Naming guidelines that apply to ADF Business Component entity objects:

- [ADFng1-04004] - **Entity object names** - Entity object names should match that of the relating database object name, but following the JDeveloper convention of using initialized camel-case names without special characters.

  For example the CUSTOMER_SALES table would become the CustomerSales entity object.

  An incorrect example would be Customer_Sales.

- [ADFng1-04005] - **Entity object attributes** - Persistent entity object attributes names should match that of the relating database column name, but following JDeveloper convention of using initialized camel-case names without special characters.

  For example the database DATE_OF_BIRTH column would become the DateOfBirth attribute name.

  Transient entity object attributes names should indicate their purpose or intent. For example a transient attribute that uses the Groovy expression Cost * Quantity could be called TotalCost.

Entity Object Associations

Naming guidelines that apply to ADF Business Component entity object associations:

- [ADFng1-04006] - **Entity object association name suffix** - Entity object associations should always have the suffix "Assoc".
- [ADFng1-04007] – **Entity object association names** – As a general rule if an entity object association is modeled on a database foreign key the name of the association should relate to the foreign key name, but following the JDeveloper convention of initialized camel-case names and dropping special characters such as underscore.

  For example the foreign key name of EMP_DEPT_FK would become the EmpDeptFkAssoc entity object association.

- [ADFng1-04008] - **Entity object logical association names** - If an entity object association is not modeled on a database foreign key but rather represents a logical relationship defined by the ADF programmer, the name of the association should end in the suffix "LogicalAssoc" to help differentiate the association from those that are based on database foreign keys.

Extensions Framework

As recommended by Oracle you should create extensions of the core ADF Business Component framework classes. The naming guidelines that apply to these classes are:

- [ADFng1-04009] - **Extension framework class names** - For every framework class you create extending the oracle.jbo.server classes, the framework class name should have a prefix of your workspace name, and a suffix of the class you are extending.

  For example if your workspace is named "Hr" and you're extending the oracle.jbo.server.EntityImpl class, then you're class should be called "HrEntityImpl".

Property Sets

Naming guidelines that apply to ADF Business Component property sets:

- [ADFng1-04010] - **Property set name suffix** - Property sets should always have the suffix "PropertySet".

  For example AuditPropertySet.

Validation Rules

Naming guidelines that apply to ADF Business Component validation rules:

- [ADFng1-04011] - **Validation rule name suffix** - Entity level validation rules should always have the suffix "EntityValidationRule". Alternatively attribute level validation rules should use the suffix "AttrValidationRule".
- [ADFng1-04012] - **Validation rule name** - The overall name of the validation rule should indicate what it's attempting to validate.

  For example EmailAttrValidationRule.

View Accessors

Naming guidelines that apply to ADF Business Component view accessors available to entity objects and view objects:

- [ADFng1-04013] - **View accessor names** - Both entity objects and view objects can reference other view objects defined as view accessors. On creating a view accessor the IDE will default its name to the view name with a numerical suffix. Remove the numerical suffix but maintain the view name. If this results in a collision in accessor names rename each view accessor such that their unique purpose is clear.

  If you define a view accessor with an optional view criteria in the View Accessors dialog, add the purpose of the view criteria to the view accessor's name.

  For example if you add a view criteria to subselect only current staff members from the StaffView, name the view accessor CurrentStaffView.

View Object Instances

Naming guidelines that apply to ADF Business Component view object instances defined within an Application Module:

- [ADFng1-04014] - **View object instance names** - An Application Module can expose the same view object multiple times. On creating a view object instance in the Application Module Wizard or Editor the IDE will default its name to the view object with a numerical suffix. Remove the numerical suffix but maintain the view object name. If this results in a collision in instance names rename each view object instance such that their unique purpose is clear.

View Object Links

Naming guidelines that apply to ADF Business Components view object links:

- [ADFng1-04015] - **View object link name suffix** - View object links should always have the suffix "Link".

- [ADFng1-04016] – **View object link name** – If a view object link is based on an entity object association, the link name should match the entity object association name but replacing the "Assoc" prefix with "Link". For example the view object link name EmpDeptFkLink is suitable for the entity object association EmpDeptFkAssoc.
- [ADFng1-04017] - **View object logical link names** - If a view object link is not modeled on an entity object association but rather represents a logical relationship defined by the ADF programmer, the name of the link should end in the suffix "LogicalLink" to help differentiate the link from those that are based on entity object associations.

View Objects

Naming guidelines for ADF Business Component view objects. Note this set of rules applies to view objects within the project, not the view object instance name as referred to through an application module.

- [ADFng1-04018] - **View object name suffix** - View object names should always have the suffix "View".
- [ADFng1-04019] - **View object names** - For updateable view objects the name should reflect the entity object they are based on. For example a view object based on the entity object Customers should be named CustomersView.

  For read only view objects the name should reflect what the view object is attempting to query.

  For example for a query to total monthly sales a suitable view object name could be MonthlySalesView.

  For view objects that include a custom where clause (not view criteria) the view object name should reflect this restriction. As example a view object that selects current staff members could use the name CurrentStaffView.
- [ADFng1-04020] - **View object attribute names** - Persistent view object attributes names based on entity object attributes should match that of the relating entity object attribute name.

  Transient view object attributes names should indicate their purpose or intent.

  For example a transient attribute that uses the Groovy expression Cost * Quantity could be called TotalCost.
- [ADFng1-04021] - **View object view criteria names** - View object criteria names should always have the suffix "ViewCriteria".

  The complete name of the view criteria should indicate what restriction the associated view criteria is placing on the parent view object.

  For example if a view criteria is designed to restrict the StaffView to only show the current staff, the view criteria could be named CurrentStaffViewCriteria.
- [ADFng1-04022] - **View object bind variable names** - The bind variables defined within a view object's query on those associated with a view criteria should be treated as variable instances when it comes to naming guidelines. Namely they should use camelcase with the first letter in lower case.

  In turn bind variable names should be prefixed with the letter "p" for parameter.

  For example pStatus or pDateOfBirth.

- [ADFng1-04023] - **View object list of value names** - View object list of values names should be prefixed with "LOV_" and then be followed by the view object attribute name they are defined against.

  For example "LOV_EmployeeId".

View Controller Naming Guidelines

The following naming guidelines are recommended for ViewController objects:

JavaServer Faces Constructs

The following sections relate to JavaServer Faces constructs such as the faces-confix.xml file, beans and more.

Beans

Naming guidelines that apply to beans:

- [ADFng2-04024] **- Java bean class name** - Do not include the scope of a bean in its Java class file name.

  A bean example which would violate this rule is CreateCustomerPageFlowScope.java.

- [ADFng1-04025] – **requestScope, backingBeanScope or viewScope bean names** - If a requestScope, backingBeanScope or viewScoped bean relates to a page or page fragment, name the bean after the page or fragment.

  For example for the ViewBookings.jspx page, the relating bean would be ViewBookings.java.

- [ADFng1-04026] - **Bounded task flow pageFlowScope bean** - If a pageFlowScope bean relates to a bounded task flow, name the bean after the bounded task flow.

  For example for a EnrolStudent.xml bounded task flow the relating bean would be EnrolStudent.java

- [ADFng2-04027] - **Bean component instant variables** - If a request or backingBean includes UI component instant variables with getters and setters, match the instant variable name and the accessors to the component ID.

  For example the following inputText field:

```
<af:inputText id="it1" binding="#{backingBeanScope.pageBackingBean.it1}"/>
```

  Note the id="it1". The resulting bean code uses the same name to be consistent:

```
public class PageBackingBean {

  private RichInputText it1;

  public void setIt1(RichInputText it1) { ... }

  public RichInputText getIt1() { ... }

}
```

- [ADFng1-04028] - **Bean methods names** - If a UI component listener has a relating method in a request or backingBean, name the method after the action type (e.g. action, selection, valueChange) and component ID.

  For example for a page called ViewBookings:

```
<af:commandButton id="Submit" actionListener="#{viewBookings.actionSubmit}".
```

Component IDs

Naming guidelines that apply to UI component IDs in pages:

- [ADFng2-04029] - **Component must have IDs** - All UI components defined in page and page fragments must have an ID if they support the ID attribute.

The following guideline has two camps of thought, those who think it will generate too much work for developers, and those who think the developers need to do more to assist QAs who require stable and meaningful component IDs for automated UI test tools like Selenium. Rather than leave the guideline out completely we'll include two versions of the guideline for you to understand the polar arguments and to make a choice yourself which one to use:

[ADFng2-04064a] - **Component ID format** – For the majority of UI component IDs generated by JDeveloper leave the ID as that generated. However for any UI component that is:

- Referred to via a partialTrigger
- Has its clientComponent=true
- Or looked up programmatically via a bean

Change the generated component ID such that is has a prefix in lowercase, drop the numeral suffix, and give the rest of the name a meaningful name in camelcase.

vs

[ADFng2-04064b] - **Component ID format** - For all UI components change the JDeveloper generated ID to use the default component ID prefix in lowercase, drop the numeral suffix, and give rest of the ID a meaningful name in camelcase. However keep the generated UI component IDs short by abbreviating the name.

The following shows an example:

```
<f:view xmlns:f="..snip.." xmlns:af="..snip..">
  <af:document title="untitled1.jsf" id="docDept">
    <af:messages id="mDept"/>
    <af:form id="fDept">
      <af:panelFormLayout id="pflDept">
        <af:inputText value="#{bindings.DepartmentId.inputValue}"
                      label="#{bindings.DepartmentId.hints.label}"
                      id="itDeptId"/>
        <af:inputText value="#{bindings.DepartmentName.inputValue}"
                      label="#{bindings.DepartmentName.hints.label}"
                      id="itDeptName"/>
```

```
          <f:facet name="footer">
            <af:commandButton text="Submit" id="cbSub"/>
          </f:facet>
        </af:panelFormLayout>
      </af:form>
    </af:document>
</f:view>
```

For reference the default JDeveloper ADF Faces components ID prefix are listed here:

af:breadCrumbs	bc	af:panelAccordian	pa
af:calendar	c	af:panelBorderLayout	pbl
af:carousel	c	af:panelBox	pb
af:carouselItem	ci	af:panelCollection	pc
af:chooseColor	cc	af:panelDashboard	pd
af:chooseDate	cd	af:panelFormLayout	pfl
af:column	c	af:panelGroupLayout	pgl
af:commandButton	cb	af:panelHeader	ph
af:commandImageLink	cil	af:panelLabelAndMessage	plam
af:commandLink	cl	af:panelList	pl
af:commandToolbarButton	ctb	af:panelSplitter	ps
af:contextInfo	ci	af:panelStretchLayout	psl
af:decorativeBox	db	af:panelTabbed	pt
af:dialog	d	af:popup	p
af:document	d	af:progressIndicator	pi
af:form	f	af:query	q
af:goButton	gb	af:quickQuery	qq
af:goImageLink	gil	af:regions	r
af:goLink	gl	af:resetButton	rb
af:goMenuItem	gmi	af:richTextEditor	rte
af:group	g	af:selectBooleanCheckbox	sbc
af:icon	i	af:selectBooleanRadio	sbr
af:image	i	af:selectManyCheckbox	smcb
af:inputColor	ic	af:selectManyChoice	smc
af:inputComboboxListOfValues	iclov	af:selectManyListbox	sml
af:inputDate	id	af:selectManyShuttle	sms
af:inputFile	if	af:selectOneChoice	soc
af:inputListOfValues	ilov	af:selectOneListbox	sol
af:inputNumberSlider	ins	af:selectOneListbox	solb
af:inputNumberSpinbox	ins	af:selectOneRadio	sor
af:inputRangeSlider	irs	af:separator	s
af:inputText	it	af:showDetail	sd
af:media	m	af:showDetailHeader	sdh
af:menu	m	af:spacer	sp
af:menuBar	mb	af:statusIndicator	si
af:message	m	af:table	t
af:messages	ms	af:toolbar	t
af:navigationPane	np	af:toolbox	t
af:outputFormatted	of	af:train	tr

af:outputLabel	ol	af:trainButtonBar	tbb
af:outputText	ot	af:tree	t
af:pageTemplate	pt		

Expression Language

Naming guidelines that apply to EL:

- [ADFng1-04030] - **Bean EL name** - When configuring a managed bean match the EL name of the bean to the class name, using camelcase with the first letter in lowercase.

  For example a pageFlowScope bean ViewBookings.java would be referred to as #{pageFlowScope.viewBookings} in EL.

Bindings

The following sections relate to files and object types that control the ADF Bindings including data controls, page definitions and similar.

Data Controls

There are currently no guidelines for Data Controls files.

Page Definitions

Naming guidelines that apply to page definition files:

- [ADFng1-04031] - **Page definition file name suffix** - Page definition files should have the suffix "PageDef".

  For example CreateEmployeePageDef.xml.

- [ADFng1-04032] - **Page definition file name for page types** - The page definition file name for page types (JSF, JSPX, JSFF) with exclusion of the suffix should match that of the relating bound page or page fragment.

  For example the page ViewDepartments.jspx should be backed by a ViewDepartmentsPageDef.xml page definition file.

- [ADFng1-04033] - **Page definition file name for task flow activities** - The page definition file name for page definitions that back data bound task flow activities should start with a prefix of the task flow file name, then the activity name, followed by the suffix defined above. Any punctuation in the file name other than underscores, such as full stops, should be converted to underscores.

  For example a method call activity names "ProcessClients" defined in the unbounded task flow represented by the adfc-config.xml file should result in a page definition file name of "adfc_config_ProcessClientsPageDef".

- [ADFng1-04034] - **Page definition file extension** - Page definitions should use the file suffix ".xml".

  For example CreateEmployeePageDef.xml

- [ADFng1-04035] - **Page definition IDs** - Within each page definition file the ID of the page definition should match the page definition file name.

- ~~[ADFng1-04036] - **Page definition DataControls.dcx usageId** - Within the DataControls.dcx file that the page definition file is referenced in, the usageId for the page should reflect the package the page definition file is stored in and the page definition name, with all fullstops replaced by underscores. e.g. view.ViewDepartments would become usageId="view_ViewDepartments".~~

Page Definition Bindings

Naming guidelines that apply to bindings within page definition files:

- [ADFng1-04037] - **Binding IDs** - All binding object IDs should be in camelcase with the first letter initcapped.

  For example EmployeesViewIterator

- [ADFng1-04038] - **Binding ID names** - All binding ID names should follow the "convention over configuration" ideal and represent what they are bound too. For example a value attribute binding mapped to the EmployeeId attribute of the EmployeesView object should have the ID "EmployeeId".
- [ADFng1-04039] - **Iterator name suffix** - All iterators should have the suffix "Iterator".

Miscellaneous File Types

The following section considers a miscellaneous set of file types used within the ViewController such as JavaScript and image files.

CSS

Naming guidelines that apply to CSS files:

- [ADFng1-04040] - **CSS file extensions** – A CSS file should always have the extension ".css".

JavaScript

Naming guidelines that apply to JavaScript files:

- [ADFng1-04041] - **JavaScript file extensions** – A JavaScript file should always have the extension ".js".

Images

There are currently no guidelines for Images files.

Resource Bundles

The following guidelines apply to resource bundles:

- [ADFng1-04042] – **Java resource bundle file extensions** – If your project uses Java class resource bundles then as per regular Java classes the file extension must be ".java".
- [ADFng1-04043] – **Property resource bundle file extensions** – If your project uses property file resource bundles then the file extension must be ".properties".
- [ADFng1-04044] – **XLIFF resource bundle file extensions** – If your project uses XLIFF resource bundles then the file extension must be ".xliff".
- [ADFnb1-04045] – **Resource bundle file names** – Ignoring the type of resource bundle you're using for your project which dictates the file extension, the file name before the extension follows this rules:

- The file name is made up of a name of your choice as a prefix, then a language and optional country codes as suffixes
- The file name prefix is your choice
- The suffix of the file name before the extension, if not the default local, should include an ISO 639 lowercase language code, and optionally an ISO 3166 uppercase country code.

For example a bundle for French Canada in either of the resource bundle types could be named:

- myBundle_fr_CA.java, myBundle_fr_CA.properties, myBundle_fr_CA.xlf

Page Templates

There are currently no guidelines for Page Template files.

Page Types

The following sections consider the different types of web pages supported in JDeveloper, namely HTML, JSPX, JSF and JSFF page fragments.

The following rules apply to all page types:

- [ADFng1-04046] - **Page name case** - All page types (HTML, JSPX, JSF, JSFF) must have files names that user camelcase with the first letter initcapped.

  For example CreateEmployees.jspx

- [ADFng1-04047] - **Page names and convention over configuration** - Apply the rules of "convention over configuration" when naming pages. For example if a page or page fragment is designed to work with the "Employees" data from ADF Business Components, ensure to have a file name that reflects this relationship such as ViewEmployees.jspx. See the next rule for page name prefixes.
- [ADFng1-04048] - **Page name prefixes** - If a page or page fragment provides a single function such as showing a table of data or an editable form comprised of fields, use a common prefix for the page/page fragment name to make it immediate obvious what the page provides, such as:
  - Table based page/fragments - prefix with "View" - e.g. ViewResources.jspx
  - Editable form page/fragments - prefix with "Edit" - e.g. EditResources.jspx
  - Graph page/fragments - prefix with "Graph" - e.g. GraphResources.jspx

HTML Pages

Naming guidelines that apply to HTML pages:

- [ADFng1-04049] - **HTML file extensions** - HTML pages should always have the extension ".html."

JSF/JSPX Pages

Naming guidelines that apply to JSF or JSPX pages:

- [ADFng1-04050] - **JSF/JSPX file extensions** - Depending on the ADF page technology used, either JSPX files in 11gR1 or a combination of JSPX or JSF pages in 11gR2 and forward, the page should have the associated file extension ".jspx" or ".jsf".

JSFF Page Fragments

Naming guidelines that apply to JSFF page fragments:

- [ADFng1-04051] - **Page fragment file extensions** - The page fragment should always have the extension ".jsff".

Task Flows

The following sections consider both unbounded and bounded task flows.

The following rules apply to all task flow types:

- [ADFng1-04052] - **Task flow activity names** - The name for page and page fragments represented in the task flow should be the same as the related file.

Activities

Naming guidelines that apply to task flow activity types:

- [ADFng1-04053] - **Data bound method call names** - When dropping data control palette operation bindings onto a task flow such as ExecuteWithParams, always rename the method call to include the object name the operation is acting on as a suffix.

  For example a Create operation on the CustomersView would become CreateCustomer.

- [ADFng1-04054] - **Task flow activity name case** - All task flow activity names should use camelcase and the first letter initcapped.

  For example ViewDepartments.

- [ADFng1-04055] - **Task flow return name** - If a task flow return is configured to Commit or Rollback the task flow return name should be Commit or Rollback accordingly.

- [ADFcc1-04056] - **Wildcard names** - Prefix wildcard navigation rules with "Global" such that they can be easily distinguished from other Control Flow Rules.[6]

Control Flow Case

Naming guidelines that apply to task flow control flow case:

- [ADFng1-04057] - **Task flow control flow case names** - The control flow case name between two activities should always reflect the activity name the navigation rule is transiting too.

  For example if there is a control flow case between two activities, and the destination activity is called ViewDepartments, a worthy control flow case name would be goViewDepartments.

- [ADFng1-04058] - **Task flow control flow case name case** - All task flow control flow case names should use camelcase with the first letter in lowercase.

  For example goViewDepartments.

---

[6] Task Flow Design Fundamentals - Section Control Flow Rules - http://bit.ly/adftaskflowfund

Exception Handlers

There are currently no guidelines for task flow exception handlers files.

Unbounded Task Flows

Naming guidelines that apply to unbounded task flows:

- [ADFng2-04059] - **Default unbounded task flow file name** - The default unbounded task flow file name should always be called adfc-config.xml

Bounded Task Flows

Naming guidelines that apply to bounded task flows:

- [ADFng1-04060] - **Task flow namespace** - Bounded task flows must define a unique namespace for their name identified uniquely by a combination of the following components:[7]

    4. Path to the taskflow definition file relative to the HTML root
    5. Name of the taskflow definition file
    6. ID of the taskflow within the definition file

  Thus a typical region binding might look something like the following:

```
<taskFlow id="addressTF1" taskFlowId="/WEB-INF/Addr/AddressTF.xml#AddressTF">
```

- [ADFng1-04061] - **Task flow parameter name prefix** - Task flow parameter names should be prefixed with a "p"

  For example pCustomerId.

- [ADFng1-04062] - **Task flow parameter name case** - Task flow parameter names with the exception of the prefix should be in camelcase with an initcapped first letter.

  For example pCustomerId.

Task Tlow Templates

Naming guidelines that apply to task flow templates:

- [ADFng1-04063] - **Task flow template activities names** - Standardize on a naming scheme for any activity, parameter or managed bean defined by a task flow template to minimize the risk of name collision and inadvertent overriding of the template definition.[8]

Trains

There are currently no guidelines for trains.

---

[7] Task Flow Design Fundamentals - Section Naming Considerations - http://bit.ly/adftaskflowfund

[8] Task Flow Design Fundamentals - Section Task Flow Templates - http://bit.ly/adftaskflowfund