

---

# WebObjects File Format Reference



2008-11-19



Apple Inc.  
© 2008 Apple Inc.  
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

Apple Inc.  
1 Infinite Loop  
Cupertino, CA 95014  
408-996-1010

Apple, the Apple logo, Mac, Mac OS, and WebObjects are trademarks of Apple Inc., registered in the United States and other countries.

Enterprise Objects is a trademark of Apple Inc.

Java and all Java-based trademarks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Times is a registered trademark of Heidelberger Druckmaschinen AG, available from Linotype Library GmbH.

Simultaneously published in the United States and Canada.

**Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.**

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.**

**Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.**

# Contents

---

## **Introduction to WebObjects File Format Reference 7**

Organization of This Document 7  
See Also 7

---

## **EOModel Bundle Format 9**

EOModel 10  
    Connection Dictionary 12  
EOEntity 12  
EOEntityIndex 15  
EOAttribute 17  
    Setting Value Types 20  
EORelationship 21  
EOFetchSpecification 24  
EOStoredProcedure 26

---

## **WOComponent Bundle Format 29**

The Java File 30  
The HTML File 30  
The WOD File 32  
The WOO File 33  
The API File 34

---

## **WOComponent HTML File Format 35**

WebObjects Namespaces and Tag Syntax 36  
Dynamically Resolved Non-WebObjects Tags 36

---

## **Document Revision History 37**

---



# Figures, Tables, and Listings

## **EOModel Bundle Format 9**

---

Figure 1	EOModel object diagram	10
Table 1	EOModel dictionary keys	10
Table 2	Connection dictionary keys	12
Table 3	EOEntity dictionary keys	13
Table 4	EOEntityIndex dictionary keys	16
Table 5	constraint attribute values	16
Table 6	indexType attribute values	17
Table 7	EOAttribute dictionary keys	17
Table 8	Numeric value types	20
Table 9	String value types	21
Table 10	Timestamp value types	21
Table 11	EORelationship dictionary keys	22
Table 12	Delete rules	24
Table 13	Join semantics	24
Table 14	EOFetchSpecification dictionary keys	25
Table 15	EOStoredProcedure dictionary keys	26

## **WOComponent Bundle Format 29**

---

Listing 1	Sample Main.java file	30
Listing 2	Sample HTML file	31
Listing 3	Sample WOD file	32
Listing 4	Sample WOO file	33
Listing 5	Sample API file	34

## **WOComponent HTML File Format 35**

---

Listing 1	Example WOComponent HTML file format	35
-----------	--------------------------------------	----



# Introduction to WebObjects File Format Reference

---

**Note:** This document was previously titled *WebObjects Bundle Reference*.

This document describes the format of WebObjects files and bundles that are loaded by WebObjects and EOF applications. For example, the EOModel bundle contains a description of the entity-relationship model that maps objects to tables and records in a database. The WOComponent bundle contains the layout and bindings of WOComponent objects used by web applications. The format of these bundles is public so third-party tools can export documents that can be loaded by WebObjects tools and applications.

You should read this document if you are developing a third-party tool to create either a EOModel or WOComponent object used by WebObjects and EOF applications.

## Organization of This Document

Each article in this document describes a specific file format.

- [“EOModel Bundle Format”](#) (page 9) describes the EOModel bundle, including the properties of EOModel and related objects.
- [“WOComponent Bundle Format”](#) (page 29) describes the WOComponent bundle including, the properties of WebObjects dynamic element and extensions element objects.
- [“WOComponent HTML File Format”](#) (page 35) describes the alternate HTML format of a WOComponent.

## See Also

The bundles described in this document are used by WebObjects applications. Refer to the following WebObjects documents that discuss the concepts and details about the classes and objects used in these bundles:

- Read *WebObjects Overview* to learn how WebObjects works and for an overview of the different technologies.
- Read *WebObjects Web Applications Programming Guide* for details on concepts and tasks, including descriptions of WOComponent objects.
- See *WebObjects Dynamic Elements Reference* and *WebObjects Extensions Reference* for details on elements used by WOComponent objects.
- See *WebObjects 5.4 Reference* for descriptions of core WebObjects and Enterprise Object Framework (EOF) classes.



# EOModel Bundle Format

---

<b>Availability</b>	Available in WebObjects 5.0 and later.
---------------------	--

An EOModel bundle contains property list representations of an EOModel object and its related objects. An EOModel object describes the mapping between classes, and tables and rows in a database. This article describes the files contained in the EOModel bundle, and the properties of objects that appear in these files. Each of the objects corresponds to a class in the Enterprise Object Framework (EOF). EOF applications load EOModel bundles to create the corresponding EOF objects at runtime.

An EOModel bundle is a directory containing specific files. The name of the directory is the name of the model with a `.eomodeld` suffix—for example, `Movies.eomodeld` is the directory name when `Movies` is the name of the model. In fact, tools should derive the name of the model from the name of the bundle.

The EOModel bundle contains the following files:

- `index.eomodeld`—property list representation of an EOModel object. There is only one `index.eomodeld` file per bundle. See “EOModel” (page 10) for a description of EOModel properties. This file is required.
- `<entityName>.plist`—property list representation of an EOEntity object. There are zero or more entity files per bundle—one `.plist` file per entity listed in the `index.eomodeld` file. See “EOEntity” (page 12) for a description of EOEntity properties.
- `<entityName>.fspec`—property list representation of an array of EOFetchSpecification objects associated with an entity. There are zero or more fetch specification files per bundle—one `.fspec` file per entity as needed—and one or more EOFetchSpecification objects in a fetch specification file. See “EOFetchSpecification” (page 24) for a description of EOFetchSpecification properties. If there are no fetch specifications for an entity, you are required to set the EOEntity `fetchSpecificationDictionary` key to an empty dictionary.
- `<entityName>.storedProcedure`—property list representation of an EOStoredProcedure object associated with a model. There are zero or more stored procedure files per bundle—one `.storedProcedure` file per stored procedure listed in the corresponding `index.eomodeld` file. See the “EOModel” (page 10) `storedProcedures` property and “EOStoredProcedure” (page 26) for details.

The following rules apply to property lists in the EOModel bundle:

- The contents of arrays appear in arbitrary order—they are not sorted.
- Property lists may contain legacy keys that should be mapped to the respective new keys.
- Tools do not need to save legacy keys when creating new files.
- Model and entity names should not contain characters that cannot be used in a filename on file systems available on Mac OS X and Mac OS X Server. See <http://developer.apple.com/technotes/tn/tn1150.html> for description of allowable filenames.

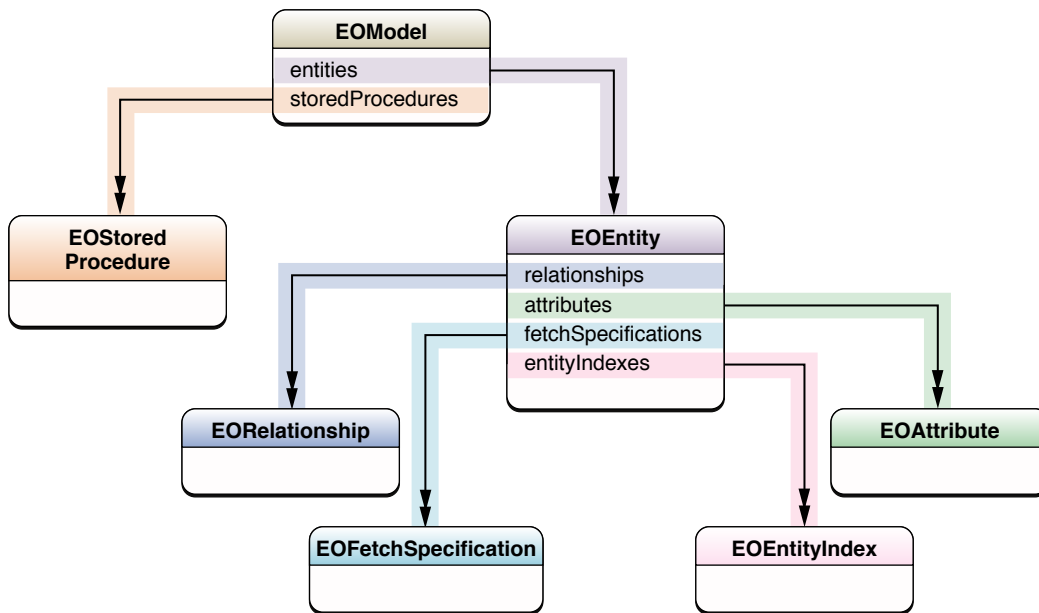
Also see *WebObjects 5.4 Reference* for more details on the corresponding EOF classes described in this document. See <http://www.apple.com/DTDs/PropertyList-1.0.dtd> for a description of the property list format supported by EOF.

## EOModel

<b>Availability</b>	Available in WebObjects 5.0 and later.
---------------------	--

An EOModel represents a mapping between a database schema and a set of classes based on the entity-relationship model for an application. The model contains a number of EOEntity objects representing the tables (entities) of the database schema. Each EOEntity object has a number of EOAttribute and EORelationship objects representing the properties (columns or fields) of the table (entity) in the database schema. In addition, EOEntity objects may have a number of EOFetchSpecification and EOStoredProcedure objects associated with them. EOEntityIndex objects are used to improve the performance of queries. Figure 1 depicts the relationship between EOModel and its components.

**Figure 1** EOModel object diagram



An EOModel object is represented as a dictionary in an `index.eomodeld` file with the property keys listed in Table 1. All keys are optional unless stated otherwise.

**Table 1** EOModel dictionary keys

Key	Type	Description
<code>EOModelVersion</code>	<code>string</code>	The version number of this model. This key is required. Available in WebObjects 5.0 and later.

Key	Type	Description
adaptorName	string	The name of the EOF adaptor used by this model—for example, JDBC. This key is required. Available in WebObjects 5.0 and later.
connectionDictionary	dictionary	A dictionary of key-value pairs used to connect to a database server—for example, initialize a JDBC or JNDI connection. See <a href="#">Table 2</a> (page 12) for a description of these key-value pairs. This key is required. Available in WebObjects 5.0 and later.
entities	array	An array of entities belonging to this model, where entities are dictionary representations of EOEntity objects with just the <code>className</code> and <code>name</code> keys set. If the <code>className</code> property is not a fully qualified Java class name, then EOF assumes the class is in the <code>com.webobjects.eocontrol</code> package. If the <code>className</code> property is a Java class that is not defined in any Java or WebObjects package, then there needs to be a corresponding <code>&lt;entityName&gt;.plist</code> file. The <code>&lt;entityName&gt;.plist</code> file must have the same values set for the <code>className</code> and <code>name</code> keys that appear in this file. See <a href="#">“EOEntity”</a> (page 12) for a description of all entity properties. This key is required. Available in WebObjects 5.0 and later.
entitiesWithShared-Objects	array	An array of entity names that specifies the entities whose objects are loaded into the the shared editing context when the application launches. An entity name is the value of an EOEntity object’s <code>name</code> key. Available in WebObjects 5.0 and later.
internalInfo	dictionary	This dictionary is for internal use by an adaptor. Available in WebObjects 5.0 and later. Deprecated in WebObjects 5.4.
storedProcedures	array	An array containing the names of all of the model’s stored procedures. See <a href="#">“EOStoredProcedure”</a> for a description of stored procedure properties. Available in WebObjects 5.0 and later.
userInfo	dictionary	A dictionary of user data that an application can use to store any auxiliary information as needed. Available in WebObjects 5.0 and later.
userDictionary	dictionary	This key is deprecated. Use the <code>userInfo</code> key instead. Available in WebObjects 5.0 and later. Deprecated in WebObjects 5.3.

## Connection Dictionary

Table 2 lists the keys that appear in a model's connection dictionary.

**Table 2** Connection dictionary keys

Key	Value
username	User name used to log in into the database. Available in WebObjects 5.0 and later.
password	Password used to log in into the database. Available in WebObjects 5.0 and later.
URL	The URL used to connect to the database of the form: <adaptor type>:<adaptor plug-in>://<database host>/<database name>. For example, set URL to jdbc:openbase://127.0.0.1/WOMovies to connect to the sample OpenBase Movies database. This key is required. Available in WebObjects 5.0 and later.
driver	The name of the database driver. (Not necessary for the adaptor plug-ins that ship with WebObjects.) Available in WebObjects 5.0 and later.
plugin	The name of the database plug-in. (Not necessary for the adaptor plug-ins that ship with WebObjects.) Available in WebObjects 5.0 and later.
adaptorName	The name of the database adaptor. For example, JDBC or JNDI. Available in WebObjects 5.0 and later.

## EOEntity

<b>Availability</b>	Available in WebObjects 5.0 and later.
---------------------	--

An EOEntity describes a table in a database and associates a name internal to the Enterprise Objects Framework with an external name by which the table is known to the database. An EOEntity maintains a group of attributes and relationships, which are collectively called properties. These are represented by the EOAttribute and EORelationship classes, respectively.

An EOEntity object is represented as a dictionary in an <entityName>.plist file with the property keys show in Table 3. An EOEntity may have multiple EOEntityIndex and EOFetchSpecification objects described in "EOEntityIndex" and "EOFetchSpecification."

**Table 3** EOEntity dictionary keys

Key	Type	Description
attributes	array	An array of attributes defining this entity where attributes are EOAttribute objects represented as dictionaries. See <a href="#">“EOAttribute”</a> (page 17) for a description of attribute properties. Available in WebObjects 5.0 and later.
attributesUsed-ForLocking	array	An array of attribute names that specifies the attributes that participate in optimistic locking. Available in WebObjects 5.0 and later.
batchFaultingMaxSize	integer	Integer greater than or equal to 0 that specifies the maximum size of batching. Available in WebObjects 5.0 and later.
cachesObjects	Boolean	Specifies whether or not the entire table is fetched into memory when the table is fetched. Caching an entity’s objects allows Enterprise Objects to evaluate queries in memory, thereby avoiding round trips to the data source. This is most useful for read-only entities where there is no danger of the cached data getting out of sync with the data in the data source. Set the value to Y to cache objects; otherwise, N. Available in WebObjects 5.0 and later.
className	string	The name of the class that corresponds to this entity. If you don’t define a custom enterprise object class for an entity, the class name defaults to EOGenericRecord. You should use a fully qualified name but this isn’t strictly required. This value should match the className value that appears in the entities array of the EModel dictionary described in Table 1. Available in WebObjects 5.0 and later.
classProperties	array	An array of attribute and relationship names that are accessible from instances of this entity. The corresponding class should provide key-value coding accessor methods for all class properties. (The EOGenericRecord class already supports key-value coding.) Any property not appearing in this array is hidden. Available in WebObjects 5.0 and later.
entityIndexes	array	EOEntityIndex objects that describe an optimization when searching on attributes. See <a href="#">“EOEntityIndex”</a> (page 15) for details. Available in WebObjects 5.4 and later.
externalName	string	The name of the table in the data source that corresponds to this entity. Available in WebObjects 5.0 and later.

Key	Type	Description
externalQuery	string	Any valid SQL statement that you want executed when unqualified fetches are performed on this entity. See <code>EOQualifier</code> in <i>WebObjects 5.4 Reference</i> for details. Available in WebObjects 5.0 and later.
fetchSpecification-Dictionary	dictionary	A dictionary representation of a fetch specification object associated with this entity. The dictionary is empty if this entity has no fetch specifications. This key is required if this entity has no fetch specifications. If this entity has one or more fetch specifications, then this key is omitted and a <code>&lt;entity-Name&gt;.fspec</code> file is used to represent an array of fetch specifications. See “ <a href="#">EOFetchSpecification</a> ” (page 24) for a description of <code>EOFetchSpecification</code> properties. Available in WebObjects 5.0 and later.
internalInfo	dictionary	This dictionary is for internal use by an adaptor. The dictionary may contain values for the <code>_javaClientClassName</code> and <code>_clientClassPropertyNames</code> keys where <code>_javaClientClassName</code> is a string and <code>_clientClassPropertyNames</code> is an array of attribute names. Available in WebObjects 5.0 and later. Deprecated in WebObjects 5.4.
isAbstractEntity	Boolean	Specifies whether or not the entity is abstract. An abstract entity is one that has no corresponding enterprise objects (no object instances) in the application. Set the value to Y if this entity is abstract; otherwise, N. Available in WebObjects 5.0 and later.
isReadOnly	Boolean	Specifies whether the data that’s represented by this entity can be altered by an application. This does not lock objects at the database level but rather works at a higher level (in the <code>EODatabaseContext</code> class) so that if you try to save changes to data that’s marked as read only, Enterprise Objects refuses the save and throws an exception. Set the value to Y if this entity is read only; otherwise, N. Available in WebObjects 5.0 and later.
maxNumberOfInstances-ToBatchFetch	integer	An integer greater than or equal to 0 that specifies the number of records—instances of this entity—fetched in each batch. Available in WebObjects 5.0 and later.
name	string	The name an application uses for this entity. Available in WebObjects 5.0 and later.
parent	string	The parent of this entity, when using inheritance. Must be the name of an <code>EOEntity</code> object. Available in WebObjects 5.0 and later.

Key	Type	Description
primaryKeyAttributes	array	An array of attribute names that are part of this entity's primary key. (Multiple attributes can represent a primary key.) Available in WebObjects 5.0 and later.
restrictingQualifier	string	Used to restrict the records that are fetched for this entity. When you add a qualifier to an entity, it invokes a fetch for that entity to retrieve objects only of the type specified by the qualifier. The value is a format string for an EOQualifier object. See EOQualifier in <i>WebObjects 5.4 Reference</i> for the format of this string. Available in WebObjects 5.0 and later.
relationships	array	An array of dictionaries where each dictionary is a representation of an EORelationship object. See Table 11 for a description of relationship properties. Available in WebObjects 5.0 and later.
sharedObjectFetch-SpecificationNames	array	An array of fetch specification names. When this EOModel is loaded, the named fetch specifications are fetched against this entity at application launch time and all fetched records are put in the application's EOWorkspace instance. Available in WebObjects 5.0 and later.
userInfo	dictionary	A dictionary of user data that an application can use to store any auxiliary information as needed. Available in WebObjects 5.0 and later.
mappingQualifier	string	This key is deprecated. Use the <code>restrictingQualifier</code> key instead. Available in WebObjects 5.0 and later. Deprecated in WebObjects 5.3.
isFetchable	Boolean	This key is deprecated. Use the <code>isAbstractEntity</code> key instead. Available in WebObjects 5.0 and later. Deprecated in WebObjects 5.3.
userDictionary	dictionary	This key is deprecated. Use the <code>userInfo</code> key instead. Available in WebObjects 5.0 and later. Deprecated in WebObjects 5.3.

## EOEntityIndex

<b>Availability</b>	Available in WebObjects 5.4 and later.
---------------------	--

An EOEntityIndex object describes an optimization when searching on attributes. For object stores that do not maintain their own automatic attribute index—typically, SQL databases—the EOEntityIndex object specifies the index the object store should maintain to optimize queries. Using entity indexes to improve performance is optional.

**Table 4** EOEntityIndex dictionary keys

Key	Type	Description
attributes	array	An array of attribute names defining this entity index. An attribute name is a string that correspond to the value of the name property of an EOAttribute object. This property is required. Available in WebObjects 5.4 and later.
constraint	string	Enumerated value specifying the constraints of the query. Possible values are described in Table 5 (page 16). The behavior of this setting depends on the database used. Available in WebObjects 5.4 and later.
indexType	string	Type of index to create. Possible values are described in Table 6 (page 17). Available in WebObjects 5.4 and later.
name	string	Unique name for this entity index. Available in WebObjects 5.4 and later.
order	string	The order of the attributes. Possible values are <code>asc</code> for ascending order and <code>desc</code> for descending order. These values are case insensitive. Available in WebObjects 5.4 and later.
userInfo	Dictionary	A dictionary of user data that an application can use to store any auxiliary information as needed. Available in WebObjects 5.4 and later.

Table 5 describes the possible values for the `constraint` attribute of an EOEntityIndex object. The values are case insensitive.

**Table 5** constraint attribute values

Value	Description
<code>distinct</code>	Creates a constraint such that all values in the index must be distinct. An error occurs if you try to add a new row with a key value that matches an existing row. This constraint does not apply to <code>NULL</code> values.
<code>fulltext</code>	Allows full-text searches in the database. Full-text searches can be created only for character-based or string-based attributes.
<code>spatial</code>	Supports OpenGIS geometry SQL extensions.

Table 6 describes the possible values for the `indexType` attribute of an EOEntityIndex object. The values are case insensitive.

**Table 6** indexType attribute values

Value	Description
clustered	The database stores together rows having the same cluster key value. Each distinct cluster key value is stored only once in each data block, regardless of the number of tables and rows in which it occurs. This saves disk space and improves performance for many database operations.
hashed	The database stores together rows that have the same hash key value. The hash value for a row is the value returned by the cluster's hash function. When you create a hash cluster, you can either specify a hash function or use the database internal hash function. Hash values are not actually stored in the cluster, although cluster key values are stored for every row in the cluster.

## EOAttribute

<b>Availability</b>	Available in WebObjects 5.0 and later.
---------------------	--

An EOAttribute represents a column, field, or property in a database table (entity) and associates an internal name with an external name or expression by which the property is known to the database. The property an EOAttribute represents may be a meaningful value, such as a salary or a name, or it may be an arbitrary value used for identification but with no real-world applicability (ID numbers and foreign keys for relationships fall into this category). An EOAttribute also maintains type information for binding values to the instance variables of objects. In addition, EOAttributes are used to represent arguments for EOStoredProcedures.

A dictionary representation of an EOAttribute contains the keys shown in Table 7.

**Table 7** EOAttribute dictionary keys

Attribute	Type	Description
adaptorValue-ConversionClassName	string	The name of the Java class used at runtime to invoke the conversion method. This value is optional. By default, the attribute value class is used. Available in WebObjects 5.4 and later.
adaptorValue-ConversionMethodName	string	The name of a method that is used at runtime to convert a custom class into one of the primitive types that the adaptor knows how to manipulate: String, Number, NSData, or NSTimestamp. Available in WebObjects 5.0 and later.
allowsNull	Boolean	Specifies whether or not this attribute can have a null value. A save operation fails if an enterprise object has a null value for an attribute that does not allow a null value. Set the value to Y if this attribute allows a null value; otherwise, N. Available in WebObjects 5.0 and later.

Attribute	Type	Description
className	string	The fully qualified Java class name of the attribute—for example, <code>java.lang.String</code> . Available in WebObjects 5.0 and later.
columnName	string	The name of the column in the data source that corresponds to this attribute. Available in WebObjects 5.0 and later.
definition	string	A key path that represents a flattened attribute from another entity. For example, if the Media entity has a to-one relationship to Event, and Event has a <code>startDate</code> attribute, you can add a <code>startDate</code> attribute to Media and specify <code>event.startDate</code> as the definition. Consequently, <code>startDate</code> in Media will be the same as <code>startDate</code> in Event. Required if this is a flattened attribute; otherwise, optional. Available in WebObjects 5.0 and later.
externalType	string	The data type of the attribute as it's understood by the data source. Available in WebObjects 5.0 and later.
factoryMethod-ArgumentType	string	The type of argument that should be passed to the factory method that is invoked by this EOAttribute object to create an attribute value for a custom class. Possible values are <code>EOFactoryMethodArgumentIsBytes</code> , <code>EOFactoryMethodArgumentIsData</code> , and <code>EOFactoryMethodArgumentIsString</code> . See EOAttribute in <i>WebObjects 5.4 Reference</i> for details. Available in WebObjects 5.0 and later. Value of <code>EOFactoryMethodArgumentIsNSString</code> is deprecated in WebObjects 5.4.
internalInfo	dictionary	This dictionary is for internal use by an adaptor. Available in WebObjects 5.0 and later. Deprecated in WebObjects 5.4.
isReadOnly	Boolean	Specifies whether or not the value of this attribute can be modified. Set the value to Y if this attribute is read only; otherwise, N. Available in WebObjects 5.0 and later.
name	string	The name of this attribute as it appears in an application and in the enterprise objects. Available in WebObjects 5.0 and later.

Attribute	Type	Description
precision	integer	An integer greater than or equal to 0 that indicates the precision of the database representation of attributes with a numeric type—for example, <code>Number</code> or <code>java.math.BigDecimal</code> . Available in WebObjects 5.0 and later.
prototypeName	string	A prototype attribute from which this attribute derives its characteristics. Available in WebObjects 5.0 and later.
readFormat	string	A format string used to appropriately format this attribute's value when it is read from the database. Available in WebObjects 5.0 and later.
scale	integer	Indicates the scale of the database representation for attributes with a numeric type—for example, <code>Number</code> or <code>java.math.BigDecimal</code> . Available in WebObjects 5.0 and later.
serverTimeZone	string	The time zone assumed for dates in the database server, or the local time zone if one hasn't been set. An <code>EOAdaptorChannel</code> automatically converts dates between the time zones used by the server and the client when fetching and saving values. This property applies only to attributes that represent dates. See the Java documentation for the <a href="#">TimeZone</a> for details. Available in WebObjects 5.0 and later.
userInfo	dictionary	A dictionary of user data that an application can use to store any auxiliary information as needed. Available in WebObjects 5.0 and later.
valueClassName	string	This key is deprecated. Use the <code>className</code> key instead. Available in WebObjects 5.0 and later. Deprecated in WebObjects 5.3.
valueFactory- MethodName	string	The name of the factory method used for creating a custom class value. Available in WebObjects 5.0 and later.
valueType	string	The format for custom value types—for example, <code>i</code> or <code>d</code> , which the JDBC adaptor uses to communicate with the data source. See “ <a href="#">Setting Value Types</a> ” (page 20) for possible values. Available in WebObjects 5.0 and later.
width	integer	An integer greater than or equal to 0 that indicates the maximum length (in bytes) for values mapped to this attribute. Available in WebObjects 5.0 and later.

Attribute	Type	Description
<code>writeFormat</code>	<code>string</code>	A format string used to format this attribute's value for writing to the database. Available in WebObjects 5.0 and later.
<code>updateFormat</code>	<code>string</code>	This key is deprecated. Use the <code>writeFormat</code> key instead. Available in WebObjects 5.0 and later. Deprecated in WebObjects 5.3.
<code>insertFormat</code>	<code>string</code>	This key is deprecated. Use the <code>writeFormat</code> key instead. Available in WebObjects 5.0 and later. Deprecated in WebObjects 5.3.
<code>selectFormat</code>	<code>string</code>	This key is deprecated. Use the <code>readFormat</code> key instead. Available in WebObjects 5.0 and later. Deprecated in WebObjects 5.3.
<code>externalName</code>	<code>string</code>	This key is deprecated. Use the <code>columnName</code> key instead. Available in WebObjects 5.0 and later. Deprecated in WebObjects 5.3.
<code>userDictionary</code>	<code>dictionary</code>	This key is deprecated. Use the <code>userInfo</code> key instead. Available in WebObjects 5.0 and later. Deprecated in WebObjects 5.3.
<code>maxLength</code>	<code>integer</code>	This key is deprecated. Use the <code>width</code> key instead. Available in WebObjects 5.0 and later. Deprecated in WebObjects 5.3.

## Setting Value Types

Table 8 describes the possible numeric value types for the EOAttribute `valueType` attribute.

**Table 8** Numeric value types

Value	Type
<code>B</code>	<code>java.math.BigDecimal</code>
<code>b</code>	<code>java.lang.Byte</code>
<code>c</code>	<code>java.lang.Boolean</code>
<code>d</code>	<code>java.lang.Double</code>
<code>f</code>	<code>java.lang.Float</code>
<code>i</code>	<code>java.lang.Integer</code>

Value	Type
l	<code>java.lang.Long</code>
s	<code>java.lang.Short</code>

Table 9 describes the possible string value types for the EOAttribute `valueType` attribute.

**Table 9** String value types

Value	Type	Description
	unknown	The JDBC adaptor uses <code>setString</code> if the text is less than the database's advertised maximum <code>varchar</code> length and uses <code>setCharacterStream</code> if it is too large. If the database fails to advertise a maximum length, the default is 256 characters.
C	char stream	The JDBC adaptor uses <code>setCharacterString</code> regardless of the text's length.
c	char trim string	The JDBC adaptor generates SQL using <code>RTRIM</code> to strip off all trailing spaces.
E	encoded bytes	The JDBC adaptor converts the text into raw UTF-8 bytes and then uses <code>setBinaryStream</code> to save them in a binary-typed column in the database.
S	string	The JDBC adaptor uses <code>setString</code> regardless of the text's length.

Table 10 describes the possible value types for the EOAttribute `valueType` attribute when the value class is `NSTimestamp`.

**Table 10** Timestamp value types

Value	Type	Description
	unknown	The JDBC adaptor uses <code>getObject</code> on the <code>java.sql.ResultSet</code> object and <code>setObject</code> on the <code>java.sql.PreparedStatement</code> object. It assumes the database can provide a value compatible with a <code>java.sql.Timestamp</code> object.
D	<code>java.util.Date</code>	The JDBC adaptor uses <code>setDate</code> and <code>getDate</code> .
M	coerce date	Use this in place of D if using Microsoft SQL Server. Supports only <code>java.sql.Date</code> .
t	<code>java.sql.Time</code>	The JDBC adaptor uses <code>getTime</code> and <code>setTime</code> .
T	<code>java.sql.Timestamp</code>	The JDBC adaptor uses <code>getTimestamp</code> and <code>setTimestamp</code> .

## EORelationship

<b>Availability</b>	Available in WebObjects 5.0 and later.
---------------------	--

An EORelationship object describes an association between two entities, based on attributes of those two entities. If you define relationships in an EOModel, the relationships defined in the database are automatically resolved as enterprise objects are fetched—the target objects of relationships are fetched along with the source objects.

A dictionary representation of an EORelationship object contains the keys show in Table 11.

**Table 11** EORelationship dictionary keys

Attribute	Type	Description
definition	string	A key path that represents a flattened relationship from another entity. Required if this is a flattened relationship; otherwise, optional. Available in WebObjects 5.0 and later.
deleteRule	string	The rule to use when the source object of a relationship is deleted. The options are nullify, cascade, deny, or no action as described in <a href="#">Table 12</a> (page 24). Available in WebObjects 5.0 and later.
destination	string	The name of an entity that is the destination of this relationship. Available in WebObjects 5.0 and later.
internalInfo	dictionary	This dictionary is for internal use by an adaptor. Available in WebObjects 5.0 and later. Deprecated in WebObjects 5.4.
isMandatory	Boolean	Indicates whether or not this relationship is mandatory. Enterprise Objects will not save an enterprise object that does not have all mandatory relationships set. Set the value to Y if this relationship is mandatory; otherwise, N. Available in WebObjects 5.0 and later.
isToMany	Boolean	Indicates whether this relationship is to-many or to-one. A to-many relationship has multiple targets or destination objects, and a to-one relationship has a single target. Set the value to Y if the relationship is to-many; otherwise, N. Available in WebObjects 5.0 and later.
joinSemantic	string	Specifies the join semantics of this relationship. Note that not all join semantics are supported by all database servers. See <a href="#">Table 13</a> (page 24) for a description of join semantics. Available in WebObjects 5.0 and later.
joins	array	An array of dictionaries representing joins. Each dictionary contains a <code>sourceAttribute</code> and <code>destinationAttribute</code> key where <code>sourceAttribute</code> , an attribute belonging to the source object, is joined with <code>destinationAttribute</code> , an attribute belonging to the destination object. Available in WebObjects 5.0 and later.

Attribute	Type	Description
name	string	The name of this relationship as it appears in an application and in the enterprise objects. Available in WebObjects 5.0 and later.
numberOfToManyFaults-ToBatchFetch	integer	An integer greater than or equal to 0 that indicates the number of to-many faults that are triggered at one time. Available in WebObjects 5.0 and later.
ownsDestination	Boolean	Indicates whether or not this relationship's source owns its destination objects. When a source object owns its destination objects, the destination objects are removed from the database when the source object is removed. Ownership implies that an owned object cannot exist without its owner. Set the value to Y if this relationship owns its destination object; otherwise, N. Available in WebObjects 5.0 and later.
propagatesPrimaryKey	Boolean	Specifies whether or not the primary key of the source entity is propagated to newly inserted objects in the destination of the relationship. That is, when inserting objects that are the destination of the relationship, this option suppresses primary key generation for the destination entity and instead uses the source object's primary key as the primary key for the newly inserted destination object. (Typically, for convenience, you propagate the primary key when the source object owns the destination object.) Set the value to Y if the relationship propagates the primary key; otherwise, N. Available in WebObjects 5.0 and later.
userInfo	dictionary	A dictionary of user data that an application can use to store any auxiliary information as needed. Available in WebObjects 5.0 and later.
dataPath	string	This key is deprecated. Use the <code>definition</code> key instead. Available in WebObjects 5.0 and later. Deprecated in WebObjects 5.3.
userDictionary	dictionary	This key is deprecated. Use the <code>userInfo</code> key instead. Available in WebObjects 5.0 and later. Deprecated in WebObjects 5.3.

Table 12 lists the delete values that can be specified for the `deleteRule` keys. The values are case insensitive.

**Table 12** Delete rules

Rule	Description
EODeleteRuleNullify	All destination objects are disassociated from the source object by removing references to them. Available in WebObjects 5.0 and later.
EODeleteRuleCascade	All objects that are the destination of this relationship are deleted. Available in WebObjects 5.0 and later.
EODeleteRuleDeny	If there are any destination objects in this relationship, refuses the deletion. Available in WebObjects 5.0 and later.
EODeleteRuleNoAction	The destination object is deleted but not back references to the source object. Available in WebObjects 5.0 and later.

Table 13 lists the join semantics that can be specified for the `joinSemantic` keys.

**Table 13** Join semantics

Semantic	Description
EOInnerJoin	Results are given only for destinations of the join relationship that have values other than <code>null</code> . Available in WebObjects 5.0 and later.
EOFullOuterJoin	Results are given for all source records, regardless of the values of the relationships. Available in WebObjects 5.0 and later.
EOLeftOuterJoin	Results preserve rows in the left (source) table, keeping them even if there's no corresponding row in the right table. Available in WebObjects 5.0 and later.
EORightOuterJoin	Results preserve rows in the right (destination) table. Available in WebObjects 5.0 and later.

## EOFetchSpecification

<b>Availability</b>	Available in WebObjects 5.0 and later.
---------------------	--

An `EOFetchSpecification` object collects the criteria needed to select and order a group of records or enterprise objects, whether from an external repository such as a relational database or an internal store such as an `EOEditingContext` object.

A dictionary representation of an `EOFetchSpecification` object contains the keys shown in Table 14.

**Table 14** EOFetchSpecification dictionary keys

Attribute	Type	Description
entityName	string	The name of the entity to be fetched. This key is required. Available in WebObjects 5.0 and later.
fetchLimit	integer	An integer greater than or equal to 0 indicating the maximum number of objects to fetch. Available in WebObjects 5.0 and later.
hints	dictionary	A dictionary of hints that an EODatabaseContext or other object can use to optimize or alter the results of a fetch. Available in WebObjects 5.0 and later.
isDeep	Boolean	Specifies whether or not this fetch specification is deep—if the fetch should include subentities of this entity. Set the value to Y if this fetch specification is deep; otherwise, N. By default, fetch specifications are deep. Available in WebObjects 5.0 and later.
locksObjects	Boolean	Specifies whether or not this fetch specification locks objects. Set the value to Y if this fetch specification locks objects; otherwise, N. By default, fetch specifications do not lock objects. Available in WebObjects 5.0 and later.
prefetching-RelationshipKeyPaths	array	An array of key paths that identifies relationships that should be fetched along with the objects specified by this fetch specification. For example, when fetching Listing objects, you can pre-fetch associated listingFeatures and suggestedCustomers relationships. This tells Enterprise Objects to retrieve listedFeatures and suggestedCustomers relationships along with the Listing itself, as opposed to creating faults for the objects in those relationships. Although prefetching increases the initial fetch cost, it can improve overall performance by reducing the number of round trips made to the data source. Available in WebObjects 5.0 and later.
promptsAfter-FetchLimit	Boolean	Specifies how Enterprise Objects should behave when the fetch limit is reached. If Y, the user is prompted about whether to continue fetching after the maximum has been reached. Otherwise, Enterprise Objects simply stops fetching when it reaches the limit. Available in WebObjects 5.0 and later.
qualifier	string	A formatted string for an EOQualifier object that indicates which records or objects the fetch specification should fetch. See EOQualifier in <i>WebObjects 5.4 Reference</i> for the format of this string. Available in WebObjects 5.0 and later.

Attribute	Type	Description
rawRowKeyPaths	array	An array of attribute key paths that should be fetched as raw data and returned as an array of dictionaries (instead of the normal result of objects). Available in WebObjects 5.0 and later.
refreshesRefetched-Objects	Boolean	A Boolean value of <i>Y</i> specifies that existing objects affected by this fetch specification should be overwritten with newly fetched values after they are updated or changed. If <i>N</i> , objects don't refresh—existing objects aren't updated when their data is refetched (the fetched data is simply discarded). Available in WebObjects 5.0 and later.
requiresAllQualifier-BindingVariables	Boolean	Specifies whether a missing value for a qualifier variable is ignored or whether Enterprise Objects requires that each qualifier variable have a value assigned to it. If <i>Y</i> , an exception is thrown during variable substitution if a missing value is present. If <i>N</i> , any qualifier expressions for which there are no variable bindings are pruned from the qualifier. Available in WebObjects 5.0 and later.
sortOrderings	array	An array of attribute names used to sort the destination objects of this relationship when fetched from the database. The attributes must be attributes of the destination object and are applied in the order as they appear in this array. Available in WebObjects 5.0 and later.
usesDistinct	Boolean	Specifies whether or not duplicate destination objects in a to-many relationship are removed after fetching. Set the value to <i>Y</i> if duplicate objects should be removed; otherwise, <i>N</i> . Available in WebObjects 5.0 and later.

## EOStoredProcedure

<b>Availability</b>	Available in WebObjects 5.0 and later.
---------------------	--

An EOStoredProcedure dictionary contains the keys listed in Table 15.

**Table 15** EOStoredProcedure dictionary keys

Attribute	Type	Description
name	string	The name of this stored procedure as it is defined in the EOModel object. Available in WebObjects 5.0 and later.

Attribute	Type	Description
externalName	string	The external name of this stored procedure that is used by the database. Available in WebObjects 5.0 and later.
userInfo	dictionary	A dictionary of user data that an application can use to store any auxiliary information as needed. Available in WebObjects 5.0 and later.
internalInfo	dictionary	This dictionary is for internal use by an adaptor. Available in WebObjects 5.0 and later. Deprecated in WebObjects 5.4.
arguments	array	An array of attribute names that are the arguments for this stored procedure. Available in WebObjects 5.0 and later.



# WOComponent Bundle Format

---

<b>Availability</b>	Available in WebObjects 5.0 and later.
---------------------	--

A WOComponent bundle contains a representation of a WOComponent object and related objects that are loaded at runtime by WebObjects applications. WOComponent objects are fundamental to how dynamic content works in WebObjects. You use components to represent webpages or partial webpages generated by your website. Components are actually templates for generating HTML pages. Components are constructed from static and dynamic elements. You use dynamic elements to bind HTML counterparts to variables and methods in your component class.

A WOComponent bundle is a directory containing specific files. The name of the directory is the name of the component with a `.wo` suffix—for example, `Main.wo` is the directory name when `Main` is the name of the component. The WOComponent bundle contains the following files:

- `[componentName].java`—contains the controllers and business logic needed to make the dynamic component work. This file is mandatory but does not reside in the bundle—it needs to be loaded by the application.
- `[componentName].html`—contains markup text plus any `webobject` elements that are resolved at runtime.
- `[componentName].wod`—contains bindings that map `webobject` elements and associated values to dynamic elements or custom components. This file is optional.
- `[componentName].woo`—contains information that describes `WODisplayGroup` objects and other information for build tools such as text encodings. This file is optional if this information is covered in the Java file.
- `[componentName].api`—used for validating bindings within a component—defines what bindings are mandatory. This file is optional and does not need to reside in the bundle.

The following sections describe the format of each of these types of files.

See *WebObjects 5.4 Reference* for more details on the corresponding WebObjects classes described in this document. See *WebObjects Dynamic Elements Reference* and *WebObjects Extensions Reference* for complete descriptions of dynamic elements that may be contained in WOComponent bundles. See *WebObjects Web Applications Programming Guide* for a description of web applications that use WOComponent bundles.

This bundle format is one of two types of formats used to represent a WOComponent—read [“WOComponent HTML File Format”](#) (page 35) for a description of the HTML file format.

## The Java File

The Java file of a WOComponent bundle contains a WOComponent subclass that implements the controllers and business logic that makes the component work. This includes declarations of WODisplayGroup objects and other variables used to bind user interface elements to application objects. Typically, bindings are made directly to enterprise objects.

Listing 1 shows the Java file for a `Main.wo` component of a Direct to Web application that presents a login panel to the user. The `Main` class inherits from `WOComponent` and defines `username` and `password` instance variables that are bound to WebObjects elements in the corresponding WOD file show in Listing 3 (page 32). The Java file also contains action methods such as `defaultPage()` and `isAssistantCheckboxVisible()` that can be bound to elements in the user interface.

### Listing 1 Sample Main.java file

```
// Created by Direct to Web's Project Builder Wizard

import com.webobjects.foundation.*;
import com.webobjects.appserver.*;
import com.webobjects.directtoweb.*;

public class Main extends WOComponent {
    public String username;
    public String password;
    public boolean wantsWebAssistant = false;

    public Main(WOContext aContext) {
        super(aContext);
    }

    public WOComponent defaultPage() {
        if (isAssistantCheckboxVisible()) {
            D2W.factory().setWebAssistantEnabled(wantsWebAssistant);
        }
        return D2W.factory().defaultPage(session());
    }

    public boolean isAssistantCheckboxVisible() {
        String s = System.getProperty("D2WebAssistantEnabled");
        return s == null || NSPropertyListSerialization.booleanForString(s);
    }
}
```

## The HTML File

The HTML file inside a WOComponent bundle presents the syntactic structure of data content. It's a regular HTML document containing one special element called `webobject`. The `webobject` element has a single required attribute called `name` which corresponds to an entry in the WOD file.

A `webobject` element shares the same basic content models as HTML, namely inline and block. When a `webobject` element poses as a block level element such as a `WOConditional` element, it contains other elements. When a `webobject` element poses as an inline element such as a `WOString` element, it does not contain any other elements. It serves as a placeholder for that inline element.

Listing 2 shows an HTML file from the `Main.wo` component of a Direct to Web application containing several `webobject` elements for the Username and Password fields in a form.

**Listing 2**      Sample HTML file

```
<html>
  <head>
    <title>Direct to Web</title>
  </head>
  <body bgcolor="#ffffff">
    <h1 align="center">Welcome to Direct to Web!</h1>
    <webobject name="LoginForm">
      <center>
        <table border="2" cellpadding="4" cellspacing="2" bgcolor="#b0b0b0">
          <tr>
            <td>
              <table border="0" cellpadding="2" cellspacing="0"
bgcolor="#b0b0b0">
                <tr>
                  <th align="right">Username: </th>
                  <td>
                    <webobject name="UsernameField"></webobject>
                  </td>
                </tr>
                <tr>
                  <th align="right">Password: </th>
                  <td>
                    <webobject name="PasswordField"></webobject>
                  </td>
                </tr>
                <webobject name="AssistantConditional">
                <tr>
                  <th align="center" colspan="2">Enable Assistant:
                    <webobject name="LACheckbox"></webobject>
                  </th>
                </tr>
                </webobject>
                <tr>
                  <td colspan="2" align="center"><webobject
name="LoginButton"></webobject>
                  </td>
                </tr>
              </table>
            </td>
          </tr>
        </table>
      </center>
    </webobject>
  </body>
</html>
```

## The WOD File

The WOD file contains bindings that map the named `webobject` elements declared in the HTML file to other WOComponent objects, dynamic elements, or variables or actions in the Java file.

Each `webobject` element contained in the HTML file must have a corresponding declaration construct in the WOD file formatted as follows:

```
CDATA_NAME: ELEMENT_TYPE {
    CDATA_ATTRIBUTE_KEY = CDATA_ATTRIBUTE_VALUE;
    [CDATA_ATTRIBUTE_KEY = CDATA_ATTRIBUTE_VALUE; ...]
}
```

where `CDATA_NAME` must match the `name` attribute of the `webobject` element in the HTML file. The possible values for `ELEMENT_TYPE` and their corresponding `CDATA_ATTRIBUTE_KEY` and `CDATA_ATTRIBUTE_VALUE` values are described in *WebObjects Dynamic Elements Reference* and *WebObjects Extensions Reference*.

Listing 3 shows a WOD file from the `Main.wo` component of a Direct to Web application that binds the `Username` and `Password` fields to the corresponding `username` and `password` variables declared in the Java file. It also binds the submit button action to the `defaultPage()` method.

### Listing 3 Sample WOD file

```
AssistantConditional: WOConditional {
    condition = isAssistantCheckboxVisible;
}

DirectToWebPlaque: WOImage {
    alt = "Direct to Web - WebObjects 4.5";
    filename = "DTW.gif";
    framework = "JavaDirectToWeb";
    height = "49";
    name = "Direct to Web - WebObjects 4.5";
    width = "196";
}

LACheckbox: WOCheckBox {
    checked = wantsWebAssistant;
}

LoginButton: WOSubmitButton {
    action = defaultPage;
    value = "Login";
}

LoginForm: WOForm {
}

PasswordField: WOPasswordField {
    value = password;
    size = "16";
}

UsernameField: WOTextField {
    value = username;
}
```

```

    size = "16";
}

```

## The WOO File

The WOO file contains information for tool developers such as the character encoding for the files within the bundle, WebObjects version information, and localization information. The file also contains information on objects that are referenced by the component. Typically, an object such as a WODisplayGroup definition reside in this file—represented as an item in an array that is the value for the `variables` key.

The format of a WOO file is as follows:

```

{      CDATA_ATTRIBUTE_KEY = CDATA_ATTRIBUTE_VALUE;      [CDATA_ATTRIBUTE_KEY =
CDATA_ATTRIBUTE_VALUE; ...] }

```

where `CDATA_ATTRIBUTE_KEY` and `CDATA_ATTRIBUTE_VALUE` are key-value pairs.

Variables such as WODisplayGroup objects might appear in an array that is the value for the `variables` key. For example, Listing 4 declares a `creatureDisplayGroup` variable that is an instance of WODisplayGroup. This example is of the `FolderViewPage.woo` component in the `WOInheritanceExample` sample code located in `/Developer/Examples/JavaWebObjects`.

### Listing 4 Sample WOO file

```

{
    "WebObjects Release" = "WebObjects 5.0";
    encoding = NSMacOSRomanStringEncoding;
    variables = {
        creatureDisplayGroup = {
            class = WODisplayGroup;
            dataSource = {
                class = EODatabaseDataSource;
                editingContext = session.defaultEditingContext;
                fetchSpecification = {
                    class = EOFetchSpecification;
                    entityName = Creature;
                    fetchLimit = 0;
                    isDeep = YES;
                };
            };
            formatForLikeQualifier = "%@";
            localKeys = ();
            numberOfObjectsPerBatch = 0;
            selectsFirstObjectAfterFetch = YES;
            sortOrdering = ();
        };
    };
}

```

## The API File

The API file is used to validate bindings made between component objects—it defines the interface of a reusable component. It specifies whether an attribute is mandatory or read only, for example. Tools may use this API validation mechanism. The WebObjects runtime does not use the API files to validate bindings.

The API file has the following format:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<wodefinitions>
  <wo class=[component class name] wocomponentcontent=[true/false]>
    <binding name=[binding name] defaults=[default value]/>
    [more bindings]
    <validation message="[binding name of a settable value]">
      <unsettable name=[binding name]/>
    </validation>
  </wo>
</wodefinitions>
```

Listing 5 shows an API file from the `Main.wo` component of a Direct to Web application.

### Listing 5 Sample API file

```
<?xml version="1.0" encoding="macintosh" standalone="yes"?>
<wodefinitions>
  <wo class="Main" wocomponentcontent="NO">
  </wo>
</wodefinitions>
```

# WOComponent HTML File Format

---

<b>Availability</b>	Available in WebObjects 5.4 and later.
---------------------	--

This article describes the HTML file format of a WOComponent that is loaded by WebObjects applications. WOComponent objects are fundamental to how dynamic content works in WebObjects. You use components to represent webpages or partial webpages generated by your website. Components are actually templates for generating HTML pages. Components are constructed from static and dynamic elements. You use dynamic elements to bind HTML counterparts to variables and methods in your component class.

This format integrates the functionality of all the files contained in the original WOComponent bundle into a single HTML file. The file must be XML compliant with WOComponent references and bindings defined inline as tagged attributes. For example, `<webobjects name="titleString"></webobjects>` is represented as `<wo:WOString value="[item.title]"/>` in the XML format. The file can have an `.html`, `.xhtml`, or `.xml` file extension.

This format also supports dynamically generated elements within non-WebObjects tags such as `<td align="top" width="[width]" ..... >.....</td>`. The `td` tag is translated into a `WOGenericContainer` using the `[width]` variable. Adding arguments to this dynamic element can be accomplished by adding specially prefixed key value attributes to the element. Specifically, the `?key="value"` binding in the original bundle is represented as `woq:key-"value"` in the HTML file.

Listing 1 shows an example of this format for representing a WOComponent. The following sections describe the details of this format.

## Listing 1 Example WOComponent HTML file format

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns:wo="http://www.apple.com/"
  <head>
    <meta name="generator" content="WebObjects 5.4">
    <meta http-equiv="Content-Type" content="application/xhtml+xml;
charset=utf-8" />
    <title>Think Movies : Home</title >
  </head>

  <body>
    <div>
      <wo:ToolbarComponent/>
      <br/>
      <br/>
      <br/>
      <wo:WOImage name="BannerImage" filename = "MoviesBanner.gif" height =
"155" width = "563"/>
    </div>
  </body>
</html>
```



**Warning:** Apple reserves the right to modify the syntax and behavior of parsing the WOComponent HTML file format in future releases.

This HTML format is an alternative format to the original bundle format described in “[WOComponent Bundle Format](#)” (page 29).

## WebObjects Namespaces and Tag Syntax

This section describes the syntax of WebObjects tags in the HTML file format. The syntax is as follows:

```
<wo:[component name] [woq:]attribute="[[^][binding value]]| binding value" ...
/>
```

This syntax has the following properties:

- WebObjects tags are prefixed with the `wo:` namespace—for example, `wo:WORepetition`.
- Dynamic bindings are surrounded by bracket characters, `[` and `]`.
- Dynamic bindings can refer to their parent bindings using the `^` character.
- Keys prefixed with the `woq:` characters are analogous to prefixing `?` in the original `.wod` file in a WOComponent bundle and are used for appending `formValue` content.

For example, this is how a `WOHyperlink` is defined in the original WOComponent bundle format:

```
Hyperlink1: WOHyperlink {
    ?movieID = movieID;
    directActionName = "DisplayMovie";
}
```

In the HTML file format, it is defined as follows:

```
<wo:wohyperlink woq:movieID="[movieID]" directActionName="DisplayMovie" />
```

## Dynamically Resolved Non-WebObjects Tags

This section describes the syntax of non-WebObjects tags that contain dynamic elements. The syntax is as follows:

```
<entity attribute="[value] | value" [woq:]key="value" attribute="value1"...../>
```

This syntax has the following properties:

- Any entity value prefixed with a `$` is translated into a `WOGenericContainer`.
- Keys prefixed with the `woq:` characters are analogous to prefixing `?` in the original `.wod` files and are used for appending `formValue` content.

# Document Revision History

---

This table describes the changes to *WebObjects File Format Reference*.

Date	Notes
2008-11-19	Minor edits throughout.
2007-10-31	Changed the syntax of the WOComponent HTML file format.
2007-07-11	Changed document title from "WebObjects Bundle Reference." Added WOComponent HTML chapter, and EOEntityIndex to EOModel Bundle.
2006-11-07	Added EOModel object diagram to the EOModel Bundle article.
2006-10-03	New document that describes the WebObjects bundle formats, specifically the EOModel bundle and WOComponent bundle.

