| | [**Overview**](http://docs.google.com/overview-summary.html) | **Package** | Class | [**Use**](http://docs.google.com/package-use.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV PACKAGE**](http://docs.google.com/org/omg/Dynamic/package-summary.html)   [**NEXT PACKAGE**](http://docs.google.com/org/omg/DynamicAny/DynAnyFactoryPackage/package-summary.html) | [**FRAMES**](http://docs.google.com/index.html?org/omg/DynamicAny/package-summary.html)    [**NO FRAMES**](http://docs.google.com/package-summary.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |

## Package org.omg.DynamicAny

Provides classes and interfaces that enable traversal of the data value associated with an any at runtime, and extraction of the primitive constituents of the data value.

**See:**

[**Description**](#3znysh7)

| **Interface Summary** | |
| --- | --- |
| [**DynAny**](http://docs.google.com/org/omg/DynamicAny/DynAny.html) | Any values can be dynamically interpreted (traversed) and constructed through DynAny objects. |
| [**DynAnyFactory**](http://docs.google.com/org/omg/DynamicAny/DynAnyFactory.html) | DynAny objects can be created by invoking operations on the DynAnyFactory object. |
| [**DynAnyFactoryOperations**](http://docs.google.com/org/omg/DynamicAny/DynAnyFactoryOperations.html) | DynAny objects can be created by invoking operations on the DynAnyFactory object. |
| [**DynAnyOperations**](http://docs.google.com/org/omg/DynamicAny/DynAnyOperations.html) | Any values can be dynamically interpreted (traversed) and constructed through DynAny objects. |
| [**DynArray**](http://docs.google.com/org/omg/DynamicAny/DynArray.html) | DynArray objects support the manipulation of IDL arrays. |
| [**DynArrayOperations**](http://docs.google.com/org/omg/DynamicAny/DynArrayOperations.html) | DynArray objects support the manipulation of IDL arrays. |
| [**DynEnum**](http://docs.google.com/org/omg/DynamicAny/DynEnum.html) | DynEnum objects support the manipulation of IDL enumerated values. |
| [**DynEnumOperations**](http://docs.google.com/org/omg/DynamicAny/DynEnumOperations.html) | DynEnum objects support the manipulation of IDL enumerated values. |
| [**DynFixed**](http://docs.google.com/org/omg/DynamicAny/DynFixed.html) | DynFixed objects support the manipulation of IDL fixed values. |
| [**DynFixedOperations**](http://docs.google.com/org/omg/DynamicAny/DynFixedOperations.html) | DynFixed objects support the manipulation of IDL fixed values. |
| [**DynSequence**](http://docs.google.com/org/omg/DynamicAny/DynSequence.html) | DynSequence objects support the manipulation of IDL sequences. |
| [**DynSequenceOperations**](http://docs.google.com/org/omg/DynamicAny/DynSequenceOperations.html) | DynSequence objects support the manipulation of IDL sequences. |
| [**DynStruct**](http://docs.google.com/org/omg/DynamicAny/DynStruct.html) | DynStruct objects support the manipulation of IDL struct and exception values. |
| [**DynStructOperations**](http://docs.google.com/org/omg/DynamicAny/DynStructOperations.html) | DynStruct objects support the manipulation of IDL struct and exception values. |
| [**DynUnion**](http://docs.google.com/org/omg/DynamicAny/DynUnion.html) | DynUnion objects support the manipulation of IDL unions. |
| [**DynUnionOperations**](http://docs.google.com/org/omg/DynamicAny/DynUnionOperations.html) | DynUnion objects support the manipulation of IDL unions. |
| [**DynValue**](http://docs.google.com/org/omg/DynamicAny/DynValue.html) | DynValue objects support the manipulation of IDL non-boxed value types. |
| [**DynValueBox**](http://docs.google.com/org/omg/DynamicAny/DynValueBox.html) | DynValueBox objects support the manipulation of IDL boxed value types. |
| [**DynValueBoxOperations**](http://docs.google.com/org/omg/DynamicAny/DynValueBoxOperations.html) | DynValueBox objects support the manipulation of IDL boxed value types. |
| [**DynValueCommon**](http://docs.google.com/org/omg/DynamicAny/DynValueCommon.html) | DynValueCommon provides operations supported by both the DynValue and DynValueBox interfaces. |
| [**DynValueCommonOperations**](http://docs.google.com/org/omg/DynamicAny/DynValueCommonOperations.html) | DynValueCommon provides operations supported by both the DynValue and DynValueBox interfaces. |
| [**DynValueOperations**](http://docs.google.com/org/omg/DynamicAny/DynValueOperations.html) | DynValue objects support the manipulation of IDL non-boxed value types. |

| **Class Summary** | |
| --- | --- |
| [**\_DynAnyFactoryStub**](http://docs.google.com/org/omg/DynamicAny/_DynAnyFactoryStub.html) | DynAny objects can be created by invoking operations on the DynAnyFactory object. |
| [**\_DynAnyStub**](http://docs.google.com/org/omg/DynamicAny/_DynAnyStub.html) | Any values can be dynamically interpreted (traversed) and constructed through DynAny objects. |
| [**\_DynArrayStub**](http://docs.google.com/org/omg/DynamicAny/_DynArrayStub.html) | DynArray objects support the manipulation of IDL arrays. |
| [**\_DynEnumStub**](http://docs.google.com/org/omg/DynamicAny/_DynEnumStub.html) | DynEnum objects support the manipulation of IDL enumerated values. |
| [**\_DynFixedStub**](http://docs.google.com/org/omg/DynamicAny/_DynFixedStub.html) | DynFixed objects support the manipulation of IDL fixed values. |
| [**\_DynSequenceStub**](http://docs.google.com/org/omg/DynamicAny/_DynSequenceStub.html) | DynSequence objects support the manipulation of IDL sequences. |
| [**\_DynStructStub**](http://docs.google.com/org/omg/DynamicAny/_DynStructStub.html) | DynStruct objects support the manipulation of IDL struct and exception values. |
| [**\_DynUnionStub**](http://docs.google.com/org/omg/DynamicAny/_DynUnionStub.html) | DynUnion objects support the manipulation of IDL unions. |
| [**\_DynValueStub**](http://docs.google.com/org/omg/DynamicAny/_DynValueStub.html) | DynValue objects support the manipulation of IDL non-boxed value types. |
| [**AnySeqHelper**](http://docs.google.com/org/omg/DynamicAny/AnySeqHelper.html) | org/omg/DynamicAny/AnySeqHelper.java . |
| [**DynAnyFactoryHelper**](http://docs.google.com/org/omg/DynamicAny/DynAnyFactoryHelper.html) | DynAny objects can be created by invoking operations on the DynAnyFactory object. |
| [**DynAnyHelper**](http://docs.google.com/org/omg/DynamicAny/DynAnyHelper.html) | Any values can be dynamically interpreted (traversed) and constructed through DynAny objects. |
| [**DynAnySeqHelper**](http://docs.google.com/org/omg/DynamicAny/DynAnySeqHelper.html) | org/omg/DynamicAny/DynAnySeqHelper.java . |
| [**DynArrayHelper**](http://docs.google.com/org/omg/DynamicAny/DynArrayHelper.html) | DynArray objects support the manipulation of IDL arrays. |
| [**DynEnumHelper**](http://docs.google.com/org/omg/DynamicAny/DynEnumHelper.html) | DynEnum objects support the manipulation of IDL enumerated values. |
| [**DynFixedHelper**](http://docs.google.com/org/omg/DynamicAny/DynFixedHelper.html) | DynFixed objects support the manipulation of IDL fixed values. |
| [**DynSequenceHelper**](http://docs.google.com/org/omg/DynamicAny/DynSequenceHelper.html) | DynSequence objects support the manipulation of IDL sequences. |
| [**DynStructHelper**](http://docs.google.com/org/omg/DynamicAny/DynStructHelper.html) | DynStruct objects support the manipulation of IDL struct and exception values. |
| [**DynUnionHelper**](http://docs.google.com/org/omg/DynamicAny/DynUnionHelper.html) | DynUnion objects support the manipulation of IDL unions. |
| [**DynValueHelper**](http://docs.google.com/org/omg/DynamicAny/DynValueHelper.html) | DynValue objects support the manipulation of IDL non-boxed value types. |
| [**FieldNameHelper**](http://docs.google.com/org/omg/DynamicAny/FieldNameHelper.html) | org/omg/DynamicAny/FieldNameHelper.java . |
| [**NameDynAnyPair**](http://docs.google.com/org/omg/DynamicAny/NameDynAnyPair.html) | org/omg/DynamicAny/NameDynAnyPair.java . |
| [**NameDynAnyPairHelper**](http://docs.google.com/org/omg/DynamicAny/NameDynAnyPairHelper.html) | org/omg/DynamicAny/NameDynAnyPairHelper.java . |
| [**NameDynAnyPairSeqHelper**](http://docs.google.com/org/omg/DynamicAny/NameDynAnyPairSeqHelper.html) | org/omg/DynamicAny/NameDynAnyPairSeqHelper.java . |
| [**NameValuePair**](http://docs.google.com/org/omg/DynamicAny/NameValuePair.html) | org/omg/DynamicAny/NameValuePair.java . |
| [**NameValuePairHelper**](http://docs.google.com/org/omg/DynamicAny/NameValuePairHelper.html) | org/omg/DynamicAny/NameValuePairHelper.java . |
| [**NameValuePairSeqHelper**](http://docs.google.com/org/omg/DynamicAny/NameValuePairSeqHelper.html) | org/omg/DynamicAny/NameValuePairSeqHelper.java . |

## Package org.omg.DynamicAny Description

Provides classes and interfaces that enable traversal of the data value associated with an any at runtime, and extraction of the primitive constituents of the data value.

An any can be passed to a program that doesn't have any static information for the type of the any (code generated for the type by an IDL compiler has not been compiled with the object implementation). As a result, the object receiving the any does not have a portable method of using it.

DynAnys enable traversal of the data value associated with an any at runtime, and extraction of the primitive constituents of the data value. This is especially helpful for writing powerful generic servers (bridges, event channels supporting filtering). Similarly, this facility enables the construction of an any at runtime, without having static knowledge of its type. This is especially helpful for writing generic clients (bridges, browsers, debuggers, user interface tools).

Any values can be dynamically interpreted (traversed) and constructed through DynAny objects. A DynAny object is associated with a data value which corresponds to a copy of the value inserted into an Any. A DynAny object may be viewed as an ordered collection of component DynAnys. For DynAnys representing a basic type, such as long, or a type without components, such as an empty exception, the ordered collection of components is empty.

Each DynAny object maintains the notion of a current position into its collection of component DynAnys. The current position is identified by an index value that runs from 0 to n-1, where *n* is the number of components. The special index value -1 indicates a current position that points nowhere. For values that cannot have a current position (such as an empty exception), the index value is fixed at -1. If a DynAny is initialized with a value that has components, the index is initialized to 0. After creation of an uninitialized DynAny (that is, a DynAny that has no value but a TypeCode that permits components), the current position depends on the type of value represented by the DynAny. (The current position is set to 0 or -1, depending on whether the new DynAny gets default values for its components.)

The iteration operations rewind, seek, and next can be used to change the current position and the current\_component operation returns the component at the current position. The component\_count operation returns the number of components of a DynAny. Collectively, these operations enable iteration over the components of a DynAny, for example, to (recursively) examine its contents.

A constructed DynAny object is a DynAny object associated with a constructed type. There is a different interface, inheriting from the DynAny interface, associated with each kind of constructed type in IDL (fixed, enum, struct, sequence, union, array, exception, and value type). A constructed DynAny object exports operations that enable the creation of new DynAny objects, each of them associated with a component of the constructed data value. As an example, a DynStruct is associated with a struct value. This means that the DynStruct may be seen as owning an ordered collection of components, one for each structure member. The DynStruct object exports operations that enable the creation of new DynAny objects, each of them associated with a member of the struct.

If a DynAny object has been obtained from another (constructed) DynAny object, such as a DynAny representing a structure member that was created from a DynStruct, the member DynAny is logically contained in the DynStruct. Calling an insert or get operation leaves the current position unchanged. Destroying a top-level DynAny object (one that was not obtained as a component of another DynAny) also destroys any component DynAny objects obtained from it. Destroying a non-top level DynAny object does nothing. Invoking operations on a destroyed top-level DynAny or any of its descendants raises OBJECT\_NOT\_EXIST. If the programmer wants to destroy a DynAny object but still wants to manipulate some component of the data value associated with it, then he or she should first create a DynAny for the component and, after that, make a copy of the created DynAny object.

The behavior of DynAny objects has been defined in order to enable efficient implementations in terms of allocated memory space and speed of access. DynAny objects are intended to be used for traversing values extracted from anys or constructing values of anys at runtime. Their use for other purposes is not recommended.

## Handling DynAny objects

Insert and get operations are necessary to handle basic DynAny objects but are also helpful to handle constructed DynAny objects. Inserting a basic data type value into a constructed DynAny object implies initializing the current component of the constructed data value associated with the DynAny object. For example, invoking insert\_boolean on a DynStruct implies inserting a boolean data value at the current position of the associated struct data value. A type is consistent for inserting or extracting a value if its TypeCode is equivalent to the TypeCode contained in the DynAny or, if the DynAny has components, is equivalent to the TypeCode of the DynAny at the current position.

Basic operations include:

* insert\_boolean, get\_boolean
* insert\_char, get\_char
* insert\_short, get\_short
* insert\_ushort, get\_ushort
* insert\_long, get\_long
* insert\_ulong, get\_ulong
* insert\_double, get\_double
* insert\_string, get\_string
* insert\_reference, get\_reference
* insert\_typecode, get\_typecode
* insert\_longlong, get\_longlong
* insert\_ulonglong, get\_ulonglong
* insert\_longdouble, get\_longdouble
* insert\_wchar, get\_wchar
* insert\_wstring, get\_wstring
* insert\_any, get\_any
* insert\_dyn\_any, get\_dyn\_any
* insert\_val, get\_val
* insert\_octet, get\_octet
* insert\_float, get\_float
* get\_value
* get\_as\_string
* get\_as\_ulong
* get\_members
* get\_members\_as\_dyn\_any
* get\_discriminator
* get\_length
* get\_elements
* get\_elements\_as\_dyn\_any
* get\_boxed\_value
* get\_boxed\_value\_as\_dyn\_any

DynAny and DynAnyFactory objects are intended to be local to the process in which they are created and used. This means that references to DynAny and DynAnyFactory objects cannot be exported to other processes, or externalized with ORB.object\_to\_string(). If any attempt is made to do so, the offending operation will raise a MARSHAL system exception. Since their interfaces are specified in IDL, DynAny objects export operations defined in the standard org.omg.CORBA.Object interface. However, any attempt to invoke operations exported through the Object interface may raise the standard NO\_IMPLEMENT exception. An attempt to use a DynAny object with the DII may raise the NO\_IMPLEMENT exception.

### Package Specification

For a precise list of supported sections of official specifications with which the Java[tm] Platform, Standard Edition 6 ORB complies, see [Official Specifications for CORBA support in Java[tm] SE 6](http://docs.google.com/CORBA/doc-files/compliance.html).

**Since:** 1.4

| | [**Overview**](http://docs.google.com/overview-summary.html) | **Package** | Class | [**Use**](http://docs.google.com/package-use.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV PACKAGE**](http://docs.google.com/org/omg/Dynamic/package-summary.html)   [**NEXT PACKAGE**](http://docs.google.com/org/omg/DynamicAny/DynAnyFactoryPackage/package-summary.html) | [**FRAMES**](http://docs.google.com/index.html?org/omg/DynamicAny/package-summary.html)    [**NO FRAMES**](http://docs.google.com/package-summary.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).