| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ORBInitializer.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 CLASS**](http://docs.google.com/org/omg/PortableInterceptor/ORBIdHelper.html)   [**NEXT CLASS**](http://docs.google.com/org/omg/PortableInterceptor/ORBInitializerOperations.html) | [**FRAMES**](http://docs.google.com/index.html?org/omg/PortableInterceptor/ORBInitializer.html)    [**NO FRAMES**](http://docs.google.com/ORBInitializer.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | METHOD | DETAIL: FIELD | CONSTR | METHOD |

## **org.omg.PortableInterceptor**

Interface ORBInitializer

**All Superinterfaces:** [IDLEntity](http://docs.google.com/org/omg/CORBA/portable/IDLEntity.html), [Object](http://docs.google.com/org/omg/CORBA/Object.html), [ORBInitializerOperations](http://docs.google.com/org/omg/PortableInterceptor/ORBInitializerOperations.html), [Serializable](http://docs.google.com/java/io/Serializable.html)

public interface **ORBInitializer**extends [ORBInitializerOperations](http://docs.google.com/org/omg/PortableInterceptor/ORBInitializerOperations.html), [Object](http://docs.google.com/org/omg/CORBA/Object.html), [IDLEntity](http://docs.google.com/org/omg/CORBA/portable/IDLEntity.html)

Facilitates interceptor registration and ORB initialization.

Interceptors are intended to be a means by which ORB services gain access to ORB processing, effectively becoming part of the ORB. Since Interceptors are part of the ORB, when ORB.init returns an ORB, the Interceptors shall have been registered. Interceptors cannot be registered on an ORB after it has been returned by a call to ORB.init.

An Interceptor is registered by registering an associated ORBInitializer object which implements the ORBInitializer interface. When an ORB is initializing, it shall call each registered ORBInitializer, passing it an ORBInitInfo object which is used to register its Interceptor.

**Registering ORB Initializers in Java**

ORBInitializers are registered via Java ORB properties.

The property names are of the form:

org.omg.PortableInterceptor.ORBInitializerClass.<Service>where <Service> is the string name of a class which implementsorg.omg.PortableInterceptor.ORBInitializerTo avoid name collisions, the reverse DNS name convention should be used. For example, if company X has three initializers, it could define the following properties:

* org.omg.PortableInterceptor.ORBInitializerClass.com.x.Init1
* org.omg.PortableInterceptor.ORBInitializerClass.com.x.Init2
* org.omg.PortableInterceptor.ORBInitializerClass.com.x.Init3

During ORB.init, these ORB properties which begin with org.omg.PortableInterceptor.ORBInitializerClass shall be collected, the <Service> portion of each property shall be extracted, an object shall be instantiated with the <Service> string as its class name, and the pre\_init and post\_init methods shall be called on that object. If there are any exceptions, the ORB shall ignore them and proceed.

***Example***

A client-side logging service written by company X, for example, may have the following ORBInitializer implementation:

package com.x.logging;  
   
 import org.omg.PortableInterceptor.Interceptor;   
 import org.omg.PortableInterceptor.ORBInitializer;   
 import org.omg.PortableInterceptor.ORBInitInfo;   
   
 public class LoggingService implements ORBInitializer {   
 void pre\_init( ORBInitInfo info ) {   
 // Instantiate the Logging Service s Interceptor.   
 Interceptor interceptor = new LoggingInterceptor();   
  
 // Register the Logging Service s Interceptor.   
 info.add\_client\_request\_interceptor( interceptor );   
 }   
   
 void post\_init( ORBInitInfo info ) {   
 // This service does not need two init points.   
 }   
 }

To run a program called MyApp using this logging service, the user could type:java -Dorg.omg.PortableInterceptor.ORBInitializerClass.com.x.Logging.LoggingService MyApp

**Notes about Registering Interceptors**

Request Interceptors are registered on a per-ORB basis.

To achieve virtual per-object Interceptors, query the policies on the target from within the interception points to determine whether they should do any work.

To achieve virtual per-POA Interceptors, instantiate each POA with a different ORB. While Interceptors may be ordered administratively, there is no concept of order with respect to the registration of Interceptors. Request Interceptors are concerned with service contexts. Service contexts have no order, so there is no purpose for request Interceptors to have an order. IOR Interceptors are concerned with tagged components. Tagged components also have no order, so there is no purpose for IOR Interceptors to have an order.

Registration code should avoid using the ORB (i.e., calling ORB.init with the provided orb\_id). Since registration occurs during ORB initialization, results of invocations on this ORB while it is in this state are undefined.

**See Also:**[ORBInitInfo](http://docs.google.com/org/omg/PortableInterceptor/ORBInitInfo.html)

| **Method Summary** | |
| --- | --- |

| **Methods inherited from interface org.omg.PortableInterceptor.**[**ORBInitializerOperations**](http://docs.google.com/org/omg/PortableInterceptor/ORBInitializerOperations.html) |
| --- |
| [post\_init](http://docs.google.com/org/omg/PortableInterceptor/ORBInitializerOperations.html#post_init(org.omg.PortableInterceptor.ORBInitInfo)), [pre\_init](http://docs.google.com/org/omg/PortableInterceptor/ORBInitializerOperations.html#pre_init(org.omg.PortableInterceptor.ORBInitInfo)) |

| **Methods inherited from interface org.omg.CORBA.**[**Object**](http://docs.google.com/org/omg/CORBA/Object.html) |
| --- |
| [\_create\_request](http://docs.google.com/org/omg/CORBA/Object.html#_create_request(org.omg.CORBA.Context,%20java.lang.String,%20org.omg.CORBA.NVList,%20org.omg.CORBA.NamedValue)), [\_create\_request](http://docs.google.com/org/omg/CORBA/Object.html#_create_request(org.omg.CORBA.Context,%20java.lang.String,%20org.omg.CORBA.NVList,%20org.omg.CORBA.NamedValue,%20org.omg.CORBA.ExceptionList,%20org.omg.CORBA.ContextList)), [\_duplicate](http://docs.google.com/org/omg/CORBA/Object.html#_duplicate()), [\_get\_domain\_managers](http://docs.google.com/org/omg/CORBA/Object.html#_get_domain_managers()), [\_get\_interface\_def](http://docs.google.com/org/omg/CORBA/Object.html#_get_interface_def()), [\_get\_policy](http://docs.google.com/org/omg/CORBA/Object.html#_get_policy(int)), [\_hash](http://docs.google.com/org/omg/CORBA/Object.html#_hash(int)), [\_is\_a](http://docs.google.com/org/omg/CORBA/Object.html#_is_a(java.lang.String)), [\_is\_equivalent](http://docs.google.com/org/omg/CORBA/Object.html#_is_equivalent(org.omg.CORBA.Object)), [\_non\_existent](http://docs.google.com/org/omg/CORBA/Object.html#_non_existent()), [\_release](http://docs.google.com/org/omg/CORBA/Object.html#_release()), [\_request](http://docs.google.com/org/omg/CORBA/Object.html#_request(java.lang.String)), [\_set\_policy\_override](http://docs.google.com/org/omg/CORBA/Object.html#_set_policy_override(org.omg.CORBA.Policy%5B%5D,%20org.omg.CORBA.SetOverrideType)) |

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ORBInitializer.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 CLASS**](http://docs.google.com/org/omg/PortableInterceptor/ORBIdHelper.html)   [**NEXT CLASS**](http://docs.google.com/org/omg/PortableInterceptor/ORBInitializerOperations.html) | [**FRAMES**](http://docs.google.com/index.html?org/omg/PortableInterceptor/ORBInitializer.html)    [**NO FRAMES**](http://docs.google.com/ORBInitializer.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | METHOD | DETAIL: FIELD | CONSTR | METHOD |

[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).