BEA WebLogic Enterprise

Java API Reference

BEA WebLogic Enterprise 4.2
Document Edition 4.2
July 1999

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Java API Reference

Document Edition	Date	Software Version
4.2	July 1999	BEA WebLogic Enterprise 4.2

Packages

com.beasys	
com.beasys.BEAWrapper	
com.beasys.Tobj	
com.beasys.TobjS	
javax.transaction	
org.omg.CosTransactions	
org.omg.Security	
org.omg.SecurityLevel1	
org.omg.SecurityLevel2	

Overview Package Class **Tree Deprecated Index Help**
PREV NEXT **FRAMES NO FRAMES**

Hierarchy For All Packages

Package Hierarchies:

com.beasys, com.beasys.BEAWrapper, com.beasys.Tobj, com.beasys.TobjS, javax.transaction, org.omg.CosTransactions, org.omg.Security, org.omg.SecurityLevel1, org.omg.SecurityLevel2

Class Hierarchy

- class java.lang.Object
  - class org.omg.Security.**AuthenticationStatus** (implements org.omg.CORBA.portable.IDLEntity)
  - class com.beasys.Tobj.**AuthType** (implements org.omg.CORBA.portable.IDLEntity)
  - class com.beasys.BEAWrapper.**Callbacks**
  - class com.beasys.TobjS.**DeactivateReasonValue** (implements org.omg.CORBA.portable.IDLEntity)
  - class com.beasys.Tobj.**LName**
  - class com.beasys.Tobj.**LNameComponent**
  - class org.omg.CORBA.portable.ObjectImpl (implements org.omg.CORBA.Object)
    - class org.omg.CORBA.DynamicImplementation
      - class org.omg.PortableServer.Servant (implements java.io.Serializable)
        - class com.beasys.**Tobj_Servant**
  - class com.beasys.Tobj.**Server** (implements java.io.Serializable)
  - class java.lang.Throwable (implements java.io.Serializable)
    - class java.lang.Exception
      - class java.io.IOException
        - class java.rmi.RemoteException
          - class javax.transaction.**HeuristicCommitException**
          - class javax.transaction.**HeuristicException**
          - class javax.transaction.**HeuristicMixedException**
          - class javax.transaction.**HeuristicRollbackException**
          - class javax.transaction.**InvalidTransactionException**
          - class javax.transaction.**TransactionRequiredException**
          - class javax.transaction.**TransactionRolledbackException**
      - class com.beasys.Tobj.**NoComponent**
      - class com.beasys.Tobj.**NotImplemented**
      - class com.beasys.BEAWrapper.**NotInRequest**
      - class com.beasys.Tobj.**NotSet**
      - class com.beasys.BEAWrapper.**ObjectAlreadyActive**
      - class com.beasys.Tobj.**OverFlow**
      - class com.beasys.BEAWrapper.**ServantAlreadyActive**
      - class org.omg.CORBA.UserException (implements org.omg.CORBA.portable.IDLEntity)

- class com.beasys.TobjS.**ActivateObjectFailed** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**ApplicationProblem** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.Tobj.**CannotProceed** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**CannotProceed** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**CreateServantFailed** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**DeactivateObjectFailed** (implements org.omg.CORBA.portable.IDLEntity)
- class org.omg.CosTransactions.**HeuristicHazard** (implements org.omg.CORBA.portable.IDLEntity)
- class org.omg.CosTransactions.**HeuristicMixed** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**IllegalOperation** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**InitializeFailed** (implements org.omg.CORBA.portable.IDLEntity)
- class org.omg.CosTransactions.**InvalidControl** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.Tobj.**InvalidDomain** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**InvalidDomain** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**InvalidInterface** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.Tobj.**InvalidName** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**InvalidName** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**InvalidObject** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**InvalidObjectId** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**InvalidServant** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**NilObject** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**NoSuchElement** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**NotFound** (implements org.omg.CORBA.portable.IDLEntity)
- class org.omg.CosTransactions.**NoTransaction** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**OrbProblem** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**OutOfMemory** (implements

org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.TobjS.**OverFlow** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.Tobj.**RegistrarNotAvailable** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.TobjS.**RegistrarNotAvailable** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.TobjS.**ReleaseFailed** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.Tobj.**RMfailed** (implements org.omg.CORBA.portable.IDLEntity)
        - class org.omg.CosTransactions.**SubtransactionsUnavailable** (implements org.omg.CORBA.portable.IDLEntity)
        - class org.omg.CosTransactions.**Unavailable** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.TobjS.**UnknownInterface** (implements org.omg.CORBA.portable.IDLEntity)
    - class com.beasys.**Tobj_Bootstrap**
    - class com.beasys.Tobj.**TP**

Interface Hierarchy

- interface org.omg.CORBA.Object
    - interface org.omg.CosTransactions.**Control**(also extends org.omg.CORBA.portable.IDLEntity)
    - interface org.omg.SecurityLevel2.**Credentials**(also extends org.omg.CORBA.portable.IDLEntity)
    - interface org.omg.CosTransactions.**Current**(also extends org.omg.CORBA.portable.IDLEntity)
        - interface com.beasys.Tobj.**TransactionCurrent**(also extends org.omg.CORBA.portable.IDLEntity, org.omg.CORBA.Object)
    - interface org.omg.SecurityLevel1.**Current**(also extends org.omg.CORBA.portable.IDLEntity)
        - interface org.omg.SecurityLevel2.**Current**(also extends org.omg.CORBA.portable.IDLEntity, org.omg.CORBA.Object)
    - interface org.omg.SecurityLevel2.**Current**(also extends org.omg.SecurityLevel1.Current, org.omg.CORBA.portable.IDLEntity)
    - interface org.omg.CosLifeCycle.**FactoryFinder**(also extends org.omg.CORBA.portable.IDLEntity)
        - interface com.beasys.Tobj.**FactoryFinder**(also extends org.omg.CORBA.portable.IDLEntity, org.omg.CORBA.Object)
    - interface com.beasys.Tobj.**FactoryFinder**(also extends org.omg.CosLifeCycle.FactoryFinder, org.omg.CORBA.portable.IDLEntity)
    - interface com.beasys.Tobj.**PrincipalAuthenticator**(also extends org.omg.CORBA.portable.IDLEntity, org.omg.SecurityLevel2.PrincipalAuthenticator)
    - interface org.omg.SecurityLevel2.**PrincipalAuthenticator**(also extends org.omg.CORBA.portable.IDLEntity)
        - interface com.beasys.Tobj.**PrincipalAuthenticator**(also extends

org.omg.CORBA.portable.IDLEntity, org.omg.CORBA.Object)
    - interface com.beasys.Tobj.**TransactionCurrent**(also extends org.omg.CosTransactions.Current, org.omg.CORBA.portable.IDLEntity)
- interface java.io.Serializable
    - interface org.omg.CORBA.portable.IDLEntity
        - interface org.omg.CosTransactions.**Control**(also extends org.omg.CORBA.Object)
        - interface org.omg.SecurityLevel2.**Credentials**(also extends org.omg.CORBA.Object)
        - interface org.omg.CosTransactions.**Current**(also extends org.omg.CORBA.Object)
            - interface com.beasys.Tobj.**TransactionCurrent**(also extends org.omg.CORBA.portable.IDLEntity, org.omg.CORBA.Object)
        - interface org.omg.SecurityLevel1.**Current**(also extends org.omg.CORBA.Object)
            - interface org.omg.SecurityLevel2.**Current**(also extends org.omg.CORBA.portable.IDLEntity, org.omg.CORBA.Object)
        - interface org.omg.SecurityLevel2.**Current**(also extends org.omg.SecurityLevel1.Current, org.omg.CORBA.Object)
        - interface org.omg.CosLifeCycle.**FactoryFinder**(also extends org.omg.CORBA.Object)
            - interface com.beasys.Tobj.**FactoryFinder**(also extends org.omg.CORBA.portable.IDLEntity, org.omg.CORBA.Object)
        - interface com.beasys.Tobj.**FactoryFinder**(also extends org.omg.CosLifeCycle.FactoryFinder, org.omg.CORBA.Object)
        - interface com.beasys.Tobj.**PrincipalAuthenticator**(also extends org.omg.CORBA.Object, org.omg.SecurityLevel2.PrincipalAuthenticator)
        - interface org.omg.SecurityLevel2.**PrincipalAuthenticator**(also extends org.omg.CORBA.Object)
            - interface com.beasys.Tobj.**PrincipalAuthenticator**(also extends org.omg.CORBA.portable.IDLEntity, org.omg.CORBA.Object)
        - interface com.beasys.Tobj.**TransactionCurrent**(also extends org.omg.CosTransactions.Current, org.omg.CORBA.Object)
- interface javax.transaction.**UserTransaction**

Package com.beasys

Class Summary	
Tobj_Bootstrap	The Tobj_Bootstrap class establishes communication between a client application and a WLE domain.
Tobj_Servant	The Tobj_Servant interface defines operations that allow a CORBA object to assist in the management of its state.

Overview Package Class **Tree Deprecated Index Help**
PREV PACKAGE **NEXT PACKAGE** **FRAMES NO FRAMES**

Hierarchy For Package com.beasys

Package Hierarchies:
All Packages

Class Hierarchy

- class java.lang.Object
  - class org.omg.CORBA.portable.ObjectImpl (implements org.omg.CORBA.Object)
    - class org.omg.CORBA.DynamicImplementation
      - class org.omg.PortableServer.Servant (implements java.io.Serializable)
        - class com.beasys.**Tobj_Servant**
  - class com.beasys.**Tobj_Bootstrap**

com.beasys

Class Tobj_Bootstrap

```
java.lang.Object
  |
  +--com.beasys.Tobj_Bootstrap
```

public class **Tobj_Bootstrap**
extends java.lang.Object

The Tobj_Bootstrap class establishes communication between a client application and a WLE domain. This class also obtains object references for the other environmental objects in the WLE domain, such as the FactoryFinder, the Interface Repository, the SecurityCurrent object, and the TransactionCurrent object.

Constructor Summary

Tobj_Bootstrap(org.omg.CORBA.ORB orb, java.lang.String address_str) This method creates the Bootstrap object.
Tobj_Bootstrap(org.omg.CORBA.ORB orb, java.lang.String address_str, java.applet.Applet applet) This method creates the Bootstrap object.

Method Summary	
void	**destroy_current**() Destroys the Current objects for the domain represented by the Bootstrap object.
static java.util.Properties	**getNativeProperties**() Returns a set of properties that need to be passed in a subsequent invocation of the org.omg.CORBA.ORB.init method.
static java.util.Properties	**getRemoteProperties**() Returns the properties needed to initialize the ORB for remote clients.
UserTransaction	**getUserTransaction**() Acquires a reference to a UserTransaction object, which may then be used to begin and terminate transactions and get information about transactions.
void	**register_callback_port**(org.omg.CORBA.Object objref) This Java method is invoked to notify the ISH of a listening port in the joint client/server.
org.omg.CORBA.Object	**resolve_initial_references**(java.lang.String id) Acquires CORBA object references for the FactoryFinder, SecurityCurrent, TransactionCurrent, and InterfaceRepository objects.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Tobj_Bootstrap

```
public Tobj_Bootstrap(org.omg.CORBA.ORB orb,
                      java.lang.String address_str)
               throws org.omg.CORBA.SystemException,
                      InvalidDomain
```

This method creates the Bootstrap object. The constructor throws the org.omg.CORBA.BAD_PARAM exception if orb is null or if address is not in a valid format.

Parameters:

`orb` - This is a pointer to the ORB object in the client. The Bootstrap object uses the string_to_object method of orb internally.

`address_str` - The address of the WebLogic Enterprise domain IIOP Server Listener/Handler. The address is specified differently depending on the type of client. There can be three types of clients, as follows:

- Remote Client

  For a description of the remote clients supported by WebLogic Enterprise systems, see the Release Notes.

  For remote clients, address specifies the network address of an IIOP Server Listener/Handler through which client applications gain access to a WebLogic Enterprise domain. The address may be specified in either of the following formats:

  ```
  "//hostname:port_number"
  "//#.#.#.#:port_number"
  ```

  In the first format, the domain finds an address for hostname using the local name resolution facilities (usually DNS). The hostname must be the local machine, and the local name resolution facilities must unambiguously resolve hostname to the address of the local machine.

  In the second example, the #.#.#.# is in dotted decimal format. In dotted decimal format, each # should be a number from 0 to 255. This dotted decimal number represents the IP address of the local machine.

  In both of the above formats, port_number is the TCP port number at which the domain process listens for incoming requests. The port_number should be a number between 0 and 65535.

  Note: The network address that is specified by programmers in the Bootstrap constructor or in TOBJADDR must exactly match the network address in the server application's UBBCONFIG file. The format of the address, as well as the capitalization, must match. If the addresses do not match, the invocation to the Bootstrap constructor will fail with the following seemingly unrelated error message:

  ```
  ERROR: Unofficial connection from client at
  ip address>
  ```

  For example, if the network address is specified as //TRIXIE:3500 in the ISL command line option string in the server application's UBBCONFIG file, specifying either //192.12.4.6:3500 or //trixie:3500 in the Bootstrap constructor or in TOBJADDR will cause the connection attempt to fail. On UNIX systems, use the uname -n command on the host system to determine the capitalization used. On Windows NT systems, see the host system's Network control panel to determine the capitalization used.

  One or more TCP/IP addresses can be specified. Multiple addresses are specified using a comma-separated list. For example:

  ```
  //m1.acme.com:3050
  //m1.acme.com:3050,//m2.acme.com:3050,//m3.acme.com:3051
  ```

  If multiple addresses are specified, the addresses are tried in order until a connection is established. Any member of an address list can be specified as a parenthesized grouping of pipe-separated network addresses. For example:

```
(//m1.acme.com:3050|//m2.acme.com:3050)
```

The WebLogic Enterprise system randomly selects one of the parenthesized addresses. This strategy distributes the load randomly across a set of listener processes. The address string can be specified either in the TOBJADDR environment variable or in the address parameter of the Tobj_Bootstrap constructor. (For information about the TOBJADDR environment variable, see the chapter that describes how to manage remote client applications in the Administration Guide.) However, the address specified in Tobj_Bootstrap always take precedence over the TOBJADDR environment variable. To use the TOBJADDR environment variable to specify an address string, you must specify an empty string in the Tobj_Bootstrap address parameter.

Note: For C++ applications, TOBJADDR is an environment variable; for Java applications, it is a property; for Java applets, it is an HTML parameter.

- Native Client

  For a native client, the address parameter in the Tobj_Bootstrap constructor must always be an empty string (not a null pointer). The native client connects to the application that is specified in the TUXCONFIG environment variable. The constructor raises the org.omg.CORBA.BAD_PARAM exception if the address is not empty.

- Server Acting As a Client

  For a server, the address parameter in the Tobj_Bootstrap constructor must always be an empty string (not a null pointer). The server always connects to the application in which it is booted. The constructor raises the org.omg.CORBA.BAD_PARAM exception if the address is not empty.

Throws:

InvalidDomain - For a remote client, raised if the Bootstrap object cannot connect to the WebLogic Enterprise domain. The address of the WebLogic Enterprise domain IIOP Server Listener/Handler is specified in the constructor that is specific to the programming language. For a native client or server, raised if the domain is not booted.

Tobj_Bootstrap

```
public Tobj_Bootstrap(org.omg.CORBA.ORB orb,
                      java.lang.String address_str,
                      java.applet.Applet applet)
               throws org.omg.CORBA.SystemException,
                      InvalidDomain
```

This method creates the Bootstrap object. The constructor throws the org.omg.CORBA.BAD_PARAM exception if orb is null or if address is not in a valid format.

Parameters:

`orb` - This is a pointer to the ORB object in the client. The Bootstrap object uses the string_to_object method of orb internally.

`address_str` - The address of the WebLogic Enterprise domain IIOP Server Listener/Handler. The address is specified differently depending on the type of client. There can be three types of clients, as follows:

- Remote Client

  For a description of the remote clients supported by WebLogic Enterprise systems, see the Release Notes.

  For remote clients, address specifies the network address of an IIOP Server Listener/Handler through which client applications gain access to a WebLogic Enterprise domain. The address may be specified in either of the following formats:

  ```
  "//hostname:port_number"
  "//#.#.#.#:port_number"
  ```

  In the first format, the domain finds an address for hostname using the local name resolution facilities (usually DNS). The hostname must be the local machine, and the local name resolution facilities must unambiguously resolve hostname to the address of the local machine.

  In the second example, the #.#.#.# is in dotted decimal format. In dotted decimal format, each # should be a number from 0 to 255. This dotted decimal number represents the IP address of the local machine.

  In both of the above formats, port_number is the TCP port number at which the domain process listens for incoming requests. The port_number should be a number between 0 and 65535.

  Note: The network address that is specified by programmers in the Bootstrap constructor or in TOBJADDR must exactly match the network address in the server application's UBBCONFIG file. The format of the address, as well as the capitalization, must match. If the addresses do not match, the invocation to the Bootstrap constructor will fail with the following seemingly unrelated error message:

  ```
  ERROR: Unofficial connection from client at
  ip address>
  ```

  For example, if the network address is specified as //TRIXIE:3500 in the ISL command line option string in the server application's UBBCONFIG file, specifying either //192.12.4.6:3500 or //trixie:3500 in the Bootstrap constructor or in TOBJADDR will cause the connection attempt to fail. On UNIX systems, use the uname -n command on the host system to determine the capitalization used. On Windows NT systems, see the host system's Network control panel to determine the capitalization used.

  One or more TCP/IP addresses can be specified. Multiple addresses are specified using a comma-separated list. For example:

  ```
  //m1.acme.com:3050
  //m1.acme.com:3050,//m2.acme.com:3050,//m3.acme.com:3051
  ```

  If multiple addresses are specified, the addresses are tried in order until a connection is established. Any member of an address list can be specified as a parenthesized

grouping of pipe-separated network addresses. For example:

```
(//m1.acme.com:3050|//m2.acme.com:3050)
```

The WebLogic Enterprise system randomly selects one of the parenthesized addresses. This strategy distributes the load randomly across a set of listener processes. The address string can be specified either in the TOBJADDR environment variable or in the address parameter of the Tobj_Bootstrap constructor. (For information about the TOBJADDR environment variable, see the chapter that describes how to manage remote client applications in the Administration Guide.) However, the address specified in Tobj_Bootstrap always take precedence over the TOBJADDR environment variable. To use the TOBJADDR environment variable to specify an address string, you must specify an empty string in the Tobj_Bootstrap address parameter.

Note: For C++ applications, TOBJADDR is an environment variable; for Java applications, it is a property; for Java applets, it is an HTML parameter.

- Native Client

  For a native client, the address parameter in the Tobj_Bootstrap constructor must always be an empty string (not a null pointer). The native client connects to the application that is specified in the TUXCONFIG environment variable. The constructor raises the org.omg.CORBA.BAD_PARAM exception if the address is not empty.

- Server Acting As a Client

  For a server, the address parameter in the Tobj_Bootstrap constructor must always be an empty string (not a null pointer). The server always connects to the application in which it is booted. The constructor raises the org.omg.CORBA.BAD_PARAM exception if the address is not empty.

`applet` - This is a pointer to the client applet. If the client applet does not explicitly pass the ISH host and port to the Bootstrap constructor, you can pass this argument, which causes the Bootstrap object to search for the TOBJADDR definition in the HTML file for the applet.

Throws:

InvalidDomain - For a remote client, raised if the Bootstrap object cannot connect to the WebLogic Enterprise domain. The address of the WebLogic Enterprise domain IIOP Server Listener/Handler is specified in the constructor that is specific to the programming language. For a native client or server, raised if the domain is not booted.

Method Detail

resolve_initial_references

```
public org.omg.CORBA.Object resolve_initial_references(java.lang.String id)
                                                throws InvalidName,
                                                       org.omg.CORBA.SystemException
```

Acquires CORBA object references for the FactoryFinder, SecurityCurrent, TransactionCurrent, and InterfaceRepository objects. For the specific object reference, invoke the narrow method. For example, for FactoryFinder, invoke the Tobj.FactoryFinder.narrow method.

Parameters:

`id` - This parameter must be one of the following:

- "FactoryFinder"
- "SecurityCurrent"
- "TransactionCurrent"
- "InterfaceRepository"

Returns:

Object that corresponds to the specified type ID.

Throws:

InvalidName - Raised if id is not one of the four names allowed (see id parameter). On the server, also raised when SecurityCurrent is passed.

NO_PERMISSION - Raised if id is TransactionCurrent or SecurityCurrent and another Bootstrap object in the client owns the Current objects.

---

register_callback_port

```
public void register_callback_port(org.omg.CORBA.Object objref)
                           throws org.omg.CORBA.SystemException
```

This Java method is invoked to notify the ISH of a listening port in the joint client/server. This method should only be used for joint client/server ORBs that do not support GIOP 1.2 bidirectional capabilities; that is, GIOP 1.0 client ORBs. For GIOP 1.0 and 1.1, the ISH only supports one listening port per joint client/server; therefore, the register_callback_port method should only be called once per connected joint client/server.

Usage Notes:

If the register_callback_port method is not invoked by the joint client/server, the callback port is not registered with the ISH, and the server defaults to assymmetric outbound IIOP.

Also, if the register_callback_port method is not invoked by the joint client/server, you must start the servers IIOP Server Listener (ISL) with the -O option. The -O option enables asymmetric outbound IIOP; otherwise, server-to-client invocations will not be allowed by the ISL/ISH.

Parameters:

`objref` - The object reference created by the client.

Throws:

org.omg.CORBA.BAD_PARAM - Raised if the object is nil or if the host contained in the object does not match the connection.

org.omg.CORBA.IMP_LIMIT - Raised if the register_callback_port method is invoked more than once.

destroy_current

```
public void destroy_current()
                    throws org.omg.CORBA.SystemException
```

Destroys the Current objects for the domain represented by the Bootstrap object. This method invalidates the Current objects for the domain represented by the Bootstrap object. After invoking the destroy_current method, the Current objects are marked as invalid. Any attempt to use the old Current objects from then on throw the exception org.omg. CORBA.BAD_INV_ORDER. Good programming practice is to release all Current objects before invoking the destroy_current method.

The destroy_current method must be invoked on the Bootstrap object for the domain that currently owns the two Current objects (Transaction and Security). This also results in an implicit invocation to log off for security and implictly rolls back any transaction that was begun by the client.

The application must invoke destroy_current() before invoking the resolve_initial_references method for TransactionCurrent or SecurityCurrent on another domain; otherwise, the resolve_initial_references method raises the org.omg.CORBA.NO_PERMISSION exception.

Throws:

NO_PERMISSION - Raised if the Bootstrap object is not the owner of the Current objects.

---

getUserTransaction

```
public UserTransaction getUserTransaction()
                                  throws org.omg.CORBA.SystemException
```

Acquires a reference to a UserTransaction object, which may then be used to begin and terminate transactions and get information about transactions. Note that an invocation to the getUserTransaction method has a side effect of invoking the resolve_initial_references("TransactionCurrent") method on the corresponding Bootstrap object. An invocation of the destroy_current method on the Bootstrap object would invalidate the UserTransaction object.

Returns:

Reference to a UserTransaction object.

Throws:

java.lang.IllegalStateException - Raised if another Bootstrap object in the client owns the Current objects.

---

getNativeProperties

```
public static java.util.Properties getNativeProperties()
```

Returns a set of properties that need to be passed in a subsequent invocation of the org.omg.CORBA.ORB.init method. This subsequent invocation causes the BEA Java ORB to be initialized. The getNativeProperties method also initializes the WebLogic Enterprise infrastructure.

The GetNativeProperties method must be invoked before any attempt is made to access any class in the org.omg.CORBA package; otherwise, errors will occur when receiving CORBA exceptions from the server.

Returns:

ORB properties that must be passed to the org.omg.CORBA.ORB.init method.

---

getRemoteProperties

```
public static java.util.Properties getRemoteProperties()
```

Returns the properties needed to initialize the ORB for remote clients.

Returns:

ORB properties that must be passed to the org.omg.CORBA.ORB.init method.

---

com.beasys
Class Tobj_Servant

```
java.lang.Object
  |
  +--org.omg.CORBA.portable.ObjectImpl
        |
        +--org.omg.CORBA.DynamicImplementation
              |
              +--org.omg.PortableServer.Servant
                    |
                    +--com.beasys.Tobj_Servant
```

public abstract class **Tobj_Servant**
extends org.omg.PortableServer.Servant

The Tobj_Servant interface defines operations that allow a CORBA object to assist in the management of its state. Every implementation skeleton generated by the IDL compiler automatically inherits from the Tobj_Servant class. The Tobj_Servant class contains two methods, activate_object and deactivate_object, that can be redefined by the programmer.

Whenever a request comes in for an inactive CORBA object, the object is activated and the activate_object method is invoked on the servant. When the CORBA object is deactivated, the deactivate_object method is invoked on the servant. The timing of deactivation is driven by the implementation's activation policy. When the deactivate_object method is invoked, the TP Framework passes in a reason code to indicate why the call was made.

Note: Tobj_Servant.activate_object and Tobj_Servant.deactivate_object are the only methods that the TP Framework guarantees will be invoked for CORBA object activation and deactivation. The servant class constructor and finalizer may or may not be invoked at activation or deactivation time. Therefore, the server-application code must not do any state handling for CORBA objects in either the constructor or finalizer of the servant class.

See Also:
Serialized Form

Constructor Summary

Tobj_Servant()	

Method Summary

void	**activate_object**(java.lang.String oid) Associates an object ID with a servant.
void	**deactivate_object**(java.lang.String oid, DeactivateReasonValue dr) Removes the association of an object ID with its servant.

Methods inherited from class org.omg.PortableServer.Servant

_all_interfaces, _default_POA, _object_id, _orb, _orb, _poa, _this_object, _this_object

Methods inherited from class org.omg.CORBA.DynamicImplementation

invoke

Methods inherited from class org.omg.CORBA.portable.ObjectImpl

_create_request, _create_request, _duplicate, _get_delegate, _get_domain_managers, _get_interface_def, _get_policy, _hash, _ids, _invoke, _is_a, _is_equivalent, _is_local, _non_existent, _release, _releaseReply, _request, _request, _servant_postinvoke, _servant_preinvoke, _set_delegate, _set_policy_override, equals, hashCode, toString

Methods inherited from class java.lang.Object

getClass, notify, notifyAll, wait, wait, wait

Constructor Detail

Tobj_Servant

public **Tobj_Servant**()

Method Detail

activate_object

```
public void activate_object(java.lang.String oid)
                     throws ActivateObjectFailed,
                            OutOfMemory
```

Associates an object ID with a servant. This method gives the application an opportunity to restore the object's state when the object is activated. The state may be restored from shared memory, from an ordinary flat file, or from a database file.

Object activation is triggered by a client invoking a method on an inactive CORBA object. This causes the Portable Object Adapter (POA) to assign a servant to the CORBA object. The activate_object method is invoked before the method invoked by the client is invoked. If the activate_object method returns successfully, that is, without raising an exception, the requested method is executed on the servant.

The activate_object and deactivate_object methods and the method invoked by the client can be used by the programmer to manage object state. The particular way these methods are used to manage object state may vary according to the needs of the application. For a discussion of how these methods might be used, see Creating Java Server Applications.

If the object is currently infected with a global transaction, the activate_object method executes within the scope of that same global transaction. It is the responsibility of the programmer of the object to check that the stored state of the object is consistent. In other words, it is up to the application code to save a persistent flag that indicates whether or not the deactivate_object method successfully saved the state of the object. That flag should be checked in the activate_object method.

Parameters:

`stroid` - Specifies the object ID in string format. The object ID uniquely identifies this instance of the class. This is the same object ID that was specified in the com.beasys.Tobj.TP.create_object_reference method for the object reference used for this invocation.

Throws:

ActivateObjectFailed - If an error occurs while executing the activate_object method, the application code should raise either a CORBA standard exception or a com.beasys.TobjS.ActivateObjectFailed exception. When an exception is raised, the TP Framework catches the exception, and the following events occur:

- The activation fails.
- The method invoked by the client is not executed.
- A com.beasys.TobjS.OBJECT_NOT_EXIST exception is raised back to the client.
- If the activate_object method is executing within a transaction and the client initiated the transaction, the transaction is *not* rolled back.
- If the TP Framework automatically initiated the transaction prior to invoking the activate_object method, the transaction is rolled back. This can occur if the transaction policy is always or if the AUTOTRAN attribute is set for the interface.

Note: For each CORBA interface, set AUTOTRAN to Yes if you want a transaction to start automatically when an operation invocation is received. Setting AUTOTRAN to Yes has no effect if the interface is already in transaction mode. For more information about AUTOTRAN, refer to the Administration Guide.

- Based on which exception is raised, a message is written to the user log (ULOG) file. For more information, see the System Messages manual.

OutOfMemory - Ran out of memory.

---

deactivate_object

```
public void deactivate_object(java.lang.String oid,
                              DeactivateReasonValue dr)
                       throws DeactivateObjectFailed
```

Removes the association of an object ID with its servant. This method gives the application an opportunity to save all or part of the object's state before the object is deactivated. The state may be saved in shared memory, in an ordinary flat file, or in a database file.

Object deactivation is initiated either by the system, depending on the activation policy of the implementation for the CORBA object, or by the application. The deactivate_object method is invoked before the CORBA object is deactivated.

Deactivation may occur after an execution of a method invoked by a client if the activation policy for the CORBA object implementation is method, or as a result of the end of transactional work if the activation policy is transaction. It may also occur as the result of server shutdown if the activation policy is transaction or process.

In addition, you can use the com.beasys.Tobj.TP.deactivateEnable method to control the deactivation of CORBA objects. You can invoke the com.beasys.Tobj.TP.deactivateEnable method inside a method of an object that has its activation policy set to process. This causes the object to be deactivated at the end of the method. The com.beasys.Tobj.TP.deactivateEnable method should not be invoked in an object with the activation policy set to method or transaction.

The activate_object and deactivate_object methods and the method invoked by the client can be used by the programmer to manage object state. The manner in which these methods are used to manage object state may vary according to the needs of the application. For a discussion of how these methods might be used, see Creating Java Server Applications.

The CORBA object is allowed to vote on the outcome of the transaction when the deactivate_object method is invoked with the DR_TRANS_COMMITTING reason code. By invoking the rollback_only method on the TransactionCurrent object, the method can force the transaction to be rolled back; otherwise, the two-phase commit algorithm continues. The transaction is not necessarily committed just because the rollback_only method is not invoked in this method. Any other CORBA object or resource manager involved in the transaction could also vote to roll back the transaction.

Parameters:

`oid` - Specifies the object ID in string format. The object ID uniquely identifies this instance of the class. This is the same object ID that was specified in the com.beasys.Tobj.TP.create_object_reference method.

`dr` - Indicates the event that caused this method to be invoked.

Throws:

- If an error occurs while executing the deactivate_object method, the application code should raise either a CORBA standard exception or a com.beasys.TobjS.DeactivateObjectFailed exception. If the deactivate_object method was invoked by the TP Framework, the TP Framework catches the exception and the following events occur:

- The object is deactivated.
- If the client initiated a transaction, the transaction is not rolled back.
- If the system automatically initiated a transaction, the transaction is rolled back.
- The client is not notified of the exception that is raised in the deactivate_object method.
- Based on which exception is raised, a message is logged to the user log (ULOG) file. See the System Messages manual for more information.

Restrictions:

Note that if the object is involved in a transaction when this method is invoked, there are restrictions on what type of processing can be done based on the reason the object is invoked. If the object was involved in a transaction, the activation policy is transaction and the reason code for the call is:

DR_TRANS_ABORTED
No invocations on any CORBA objects are allowed in the method. No tpcall() is allowed. Transactions cannot be suspended or begun.

DR_TRANS_COMMITTING
No invocations on any CORBA objects are allowed in the method. No tpcall() is allowed. Transactions cannot be suspended or begun.

The reason for these restrictions is that the deactivation of objects with activation policy transaction is controlled by a call to the TP Framework from the transaction manager for the transaction. When the call with reason code DR_TRANS_COMMITTING is made, the transaction manager is executing phase 1 (prepare) of the two-phase commit. At this stage, it is not possible to issue a call to either suspend a transaction or initiate a new transaction.

Invocations on other CORBA objects are not allowed because an invocation to a CORBA object that is in another process would require that the process join the transaction; since the transaction manager is already executing the prepare phase, an error would occur. In addtion, invocations are not allowed because an invocation to a CORBA object that had no transactional properties would require that the current transaction be suspended, which would cause an error. A tpcall() is not allowed for the same reason.

Similarly, when the invocation with reason code DR_TRANS_ABORTED is made, the transaction manager is already aborting. While the transaction manager is aborting, it is not possible to either suspend a transaction or initiate a new transaction. The same restrictions apply as for DR_TRANS_COMMITTING.

Package com.beasys.BEAWrapper

Class Summary	
Callbacks	Supplies a set of methods that provide similar functionality as the methods provided by the C++ wrappers.

Exception Summary	
NotInRequest	The method was invoked when the ORB was not in the context of a request; that is, not while servicing a request in method code.
ObjectAlreadyActive	The stringified object ID is already being used for a callback.
ServantAlreadyActive	The servant is already being used for a callback.

Hierarchy For Package com.beasys.BEAWrapper

Package Hierarchies:
All Packages

Class Hierarchy

- class java.lang.Object
  - class com.beasys.BEAWrapper.**Callbacks**
  - class java.lang.Throwable (implements java.io.Serializable)
    - class java.lang.Exception
      - class com.beasys.BEAWrapper.**NotInRequest**
      - class com.beasys.BEAWrapper.**ObjectAlreadyActive**
      - class com.beasys.BEAWrapper.**ServantAlreadyActive**

com.beasys.BEAWrapper
Class Callbacks

```
java.lang.Object
  |
  +--com.beasys.BEAWrapper.Callbacks
```

public class **Callbacks**
extends java.lang.Object

Supplies a set of methods that provide similar functionality as the methods provided by the C++ wrappers. These routines provide a POA-like functionality in terms of defining the LifeSpan policies of object references created by the provided methods.

Constructor Summary

`Callbacks``()` Returns a reference to the Callbacks interface.
`Callbacks``(org.omg.CORBA.ORB init_orb)` Returns a reference to the Callbacks interface.

Method Summary	
java.lang.String	**get_string_oid**() Returns the string version of the object ID of the current request.
org.omg.CORBA.Object	**restart_persistent_systemid**(org.omg.CORBA.portable.ObjectImpl servant, java.lang.String rep_id, java.lang.String stroid) Performs the following actions: Activates an object using the servant supplied to service objects of the type rep_id, using the supplied stroid (a stringified object ID), which must have been obtained by a previous invocation on the start_persistent_systemid method.
org.omg.CORBA.Object	**start_persistent_systemid**(org.omg.CORBA.portable.ObjectImpl servant, java.lang.String rep_id, org.omg.CORBA.StringHolder stroid) Performs the following actions: Activates an object using the servant supplied to service objects of the type rep_id, using an object ID generated by the system.
org.omg.CORBA.Object	**start_persistent_userid**(org.omg.CORBA.portable.ObjectImpl servant, java.lang.String rep_id, java.lang.String stroid) Performs the following actions: Activates an object using the servant supplied to service objects of the type rep_id, using the object ID stroid.
org.omg.CORBA.Object	**start_transient**(org.omg.CORBA.portable.ObjectImpl servant, java.lang.String rep_id) Performs the following actions: Activates an object using the servant supplied to service objects of the type rep_id, using an object ID generated by the system.
void	**stop_all_objects**() Tells the ORB to stop accepting requests on all servants, if any, that have been set up in this JVM.
void	**stop_object**(org.omg.CORBA.portable.ObjectImpl servant) Tells the ORB to stop accepting requests on the given servant.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Callbacks

public **Callbacks**()

Returns a reference to the Callbacks interface. Only one such object should be created for a Java Virtual Machine (JVM), even if multiple threads are used. Using more than one such object will result in unpredictable behavior.

Throws:

org.omg.CORBA.IMP_LIMIT - The BEAWrapper.Callbacks class has already been instantiated with an ORB reference. Only one instance of this class can be used in a JVM.

Callbacks

```
public Callbacks(org.omg.CORBA.ORB init_orb)
```

Returns a reference to the Callbacks interface. Only one such object should be created for a JVM, even if multiple threads are used. Using more than one such object will result in unpredictable behavior.

Parameters:

`init_orb` - The ORB to be used for all further operations.

Throws:

org.omg.CORBA.IMP_LIMIT - The BEAWrapper.Callbacks class has already been instantiated with an ORB reference. Only one instance of this class can be used in a JVM.

Method Detail

start_transient

```
public org.omg.CORBA.Object start_transient(org.omg.CORBA.portable.ObjectImpl servant,
                                            java.lang.String rep_id)
                                     throws ServantAlreadyActive
```

Performs the following actions:

- Activates an object using the servant supplied to service objects of the type `rep_id`, using an object ID generated by the system.
- Sets the ORB into the state in which it accepts requests on that object.
- Return an object reference to the object so activated. The object reference returned will be valid only until the termination of the client or until the callback servant is halted by the user via the `stop_object` method; after that, invocations on that object reference are no longer valid and can never be made valid.

Parameters:

`servant` - An instance of the Java implementation class for the interface.

`rep_id` - The repository ID of the interface.

Returns:

A reference to the servant object that was created with the object ID generated by the system and the `rep_id` provided by the user. The object reference will need to be converted to a specific object type by invoking the _narrow() operation defined for the specific object.

Throws:

ServantAlreadyActive - The servant is already being used for a callback. A servant can be used only for a callback with a single object ID. To receive callbacks on objects containing different object IDs, you must create different servants and activate them separately. The same servant can be re-used only if a stop_object operation tells the system to stop using the servant for its original object ID.

BAD_PARAMETER - The repository ID was a null string or the servant was a null pointer.

---

start_persistent_systemid

```
public org.omg.CORBA.Object start_persistent_systemid(org.omg.CORBA.portable.ObjectImpl servant,
                                                       java.lang.String rep_id,
                                                       org.omg.CORBA.StringHolder stroid)
                                                throws ServantAlreadyActive
```

Performs the following actions:

- Activates an object using the servant supplied to service objects of the `type rep_id`, using an object ID generated by the system.
- Sets the ORB into the state in which it accepts requests on that object.
- Sets the output parameter `stroid` to the stringified version of an object ID assigned by the system.
- Returns an object reference to the activated object. The object reference returned is valid even after termination of the client. That is, if the client terminates, restarts again, and then activates a servant with the same `rep_id` and for the same object ID, the servant accepts requests made on that same object reference.

Parameters:

`servant` - Is an instance of the JAVA implementation class for the interface.

`rep_id` - Is the repository ID of the interface.

`stroid` - Is set by the system and is opaque to the user. The client uses it when the client reactivates the object at a later time (using the restart_persistent_systemid method), most likely after the client process has terminated and restarted. If an exception occurs, the value returned is invalid and is set to a null string.

Returns:

An object reference created with the object ID generated by the system and the rep_id provided by the user. The object reference needs to be converted to a specific object type by invoking the _narrow() operation defined for the specific object. The caller is responsible for initializing object state and saving to persistent store, if necessary, when it is finished execution.

Throws:

ServantAlreadyActive - The servant is already being used for a callback. A servant can be used only for a callback with a single object ID. To receive callbacks on objects containing different object IDs, you must create different servants and activate them separately. The same servant can be reused only if a stop operation tells the system to stop using the servant for its original object ID.

BAD_PARAMETER - The repository ID was a null string or the servant was a null pointer.

org.omg.CORBA.IMP_LIMIT - In addition to other system reasons for this exception, a reason unique to this situation is that the JointClientServer ORB was not initialized with a port number, so that a persistent object reference cannot be created.

restart_persistent_systemid

```
public org.omg.CORBA.Object restart_persistent_systemid(org.omg.CORBA.portable.ObjectImpl servant,
                                                        java.lang.String rep_id,
                                                        java.lang.String stroid)
                                                 throws ServantAlreadyActive,
                                                        ObjectAlreadyActive
```

Performs the following actions:

- Activates an object using the servant supplied to service objects of the type rep_id, using the supplied stroid (a stringified object ID), which must have been obtained by a previous invocation on the start_persistent_systemid method.
- Sets the ORB into the state in which it accepts requests on that object.
- Returns an object reference to the activated object.

Parameters:

`servant` - Is an instance of the Java implementation class for the interface.

`rep_id` - Is the repository ID of the interface.

`stroid` - Is the stringified version of the object ID provided by the user to be set in the object reference being created. This argument must have been returned from a previous invocation to the start_persistent_systemid method.

Returns:

An object reference created with the stringified object ID `stroid` and the rep_id provided by the user. The object reference will need to be converted to a specific object type by invoking the _narrow() operation defined for the specific object. The caller is responsible for initializing object state and saving to persistent store, if necessary, when it is finished execution.

Throws:

ObjectAlreadyActive - The stringified object ID is already being used for a callback. A given object ID can have only one servant associated with it. If you want to change to a different servant, you must first invoke the stop_object method with the servant currently in use.

BAD_PARAMETER - The repository ID was a null string, the servant was a null pointer, or the object ID supplied was not previously assigned by the system.

org.omg.CORBA.IMP_LIMIT - In addition to other system reasons for this exception, the JointClientServer ORB was not initialized with a port number; therefore, a persistent object reference cannot be created.

---

start_persistent_userid

```
public org.omg.CORBA.Object start_persistent_userid(org.omg.CORBA.portable.ObjectImpl servant,
                                                    java.lang.String rep_id,
                                                    java.lang.String stroid)
                                             throws ServantAlreadyActive,
                                                    ObjectAlreadyActive
```

Performs the following actions:

- Activates an object using the servant supplied to service objects of the type `rep_id`, using the object ID stroid.
- Sets the ORB into the state in which it accepts requests on that object.
- Returns an object reference to the activated object. The object reference returned is valid even after termination of the client. That is, if the client terminates, and restarts again, and then activates a servant with the same rep_id and for the same object ID, the servant accepts requests made on that same object reference.

Parameters:

`servant` - Is an instance of the Java implementation class for the interface.

`rep_id` - Is the repository ID of the interface.

`stroid` - Is the stringified version of an object ID provided by the user to be set in the object reference being created. The `stroid` holds application-specific data and is opaque to the ORB.

Returns:

An object reference created with the stringified object ID `stroid` and the `rep_id` provided by the user. The object reference will need to be converted to a specific object type by invoking the _narrow() operation defined for the specific object. The caller is responsible for initializing object state and saving to persistent store, if necessary, when the object is finished execution.

Throws:

ServantAlreadyActive - The servant is already being used for a callback. A servant can be used only for a callback with a single object ID. To receive callbacks on objects containing different object IDs, you must create different servants and activate them separately. The same servant can be reused only if a `stop_object` operation tells the system to stop using the servant for its original object ID.

ObjectAlreadyActive - The stringified object ID is already being used for a callback. A given object ID can have only one servant associated with it. If you want to change to a different servant, you must first invoke the `stop_object` method with the servant currently in use.

BAD_PARAMETER - The repository ID was a null string or the servant was a null pointer.

org.omg.CORBA.IMP_LIMIT - In addition to other system reasons for this exception, the JointClientServer ORB was not initialized with a port number; therefore, a persistent object reference cannot be created.

---

stop_object

```
public void stop_object(org.omg.CORBA.portable.ObjectImpl servant)
```

Tells the ORB to stop accepting requests on the given servant. If the servant was not already activated, no error is reported.

Parameters:

`servant` - Is an instance of the Java implementation class for the interface. The association between this servant and its object ID will be removed from the Active Object Map.

---

get_string_oid

```
public java.lang.String get_string_oid()
                                throws NotInRequest
```

Returns the string version of the object ID of the current request. This method should be invoked only in servant code. Calling this method in client thread will throw the NotInRequest exception because the client thread has no OID associated to it.

Returns:

The string version of the object ID of the current request. This is the string that was supplied when the object reference was created. The string is meaningful to users only in the case when the object reference was created by the start_persistent_userid method. (That is, the object ID created by the `start_transient` and `start_persistent_systemid` methods were created by the ORB and have no relationship to the user application.)

Throws:

NotInRequest - The function was called when the ORB was not in the context of a request; that is, not while servicing a request in method code. Do not invoke this function from client code. Invoking this method is valid only during the execution of a method of the WLE Object (that is, the servant).

---

stop_all_objects

```
public void stop_all_objects()
```

Tells the ORB to stop accepting requests on all servants, if any, that have been set up in this JVM.

---

com.beasys.BEAWrapper
Class NotInRequest

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--com.beasys.BEAWrapper.NotInRequest
```

public final class **NotInRequest**
extends java.lang.Exception

The method was invoked when the ORB was not in the context of a request; that is, not while servicing a request in method code. Do not invoke this method from client code. Invoking this method is valid only during the execution of a method of the WLE Object (that is, the servant).

See Also:
Serialized Form

Constructor Summary

NotInRequest()

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

NotInRequest

```
public NotInRequest()
```

---

com.beasys.BEAWrapper
Class ObjectAlreadyActive

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--com.beasys.BEAWrapper.ObjectAlreadyActive
```

public final class **ObjectAlreadyActive**
extends java.lang.Exception

The stringified object ID is already being used for a callback. A given object ID can have only one servant associated with it. If you want to change to a different servant, you must first invoke the stop_object method with the servant currently in use.

See Also:
Serialized Form

Constructor Summary

ObjectAlreadyActive()

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

ObjectAlreadyActive

public **ObjectAlreadyActive**()

---

com.beasys.BEAWrapper
Class ServantAlreadyActive

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--com.beasys.BEAWrapper.ServantAlreadyActive
```

public final class **ServantAlreadyActive**
extends java.lang.Exception

The servant is already being used for a callback. A servant can be used only for a callback with a single `ObjectId`. To receive callbacks on objects containing different `ObjectIds`, you must create different servants and activate them separately. The same servant can be reused only if a stop_object operation tells the system to stop using the servant for its original `ObjectId`.

See Also:
Serialized Form

Constructor Summary

`ServantAlreadyActive``()`

Methods inherited from class java.lang.Throwable
`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

ServantAlreadyActive

```
public ServantAlreadyActive()
```

Package com.beasys.Tobj

Interface Summary	
FactoryFinder	The FactoryFinder interface provides clients with one object reference that serves as the single point of entry into the M3 domain.
PrincipalAuthenticator	The PrincipalAuthenticator interface is used to log on to and log off of the M3 domain.
TransactionCurrent	The TransactionCurrent interface supports all the methods of the Current interface in the CosTransactions module, as described in the previous section.

Class Summary	
AuthType	The type of authentication expected by the M3 domain.
LName	This class implements the names library in CosNaming.
LNameComponent	This class implements the names library in CosNaming.
Server	Provides callback methods that can be used for application-specific server initialization and termination logic.
TP	Supplies a set of service methods that can be invoked by application code.

Exception Summary	
CannotProceed	This exception is raised if the FactoryFinder or the CORBAservices Naming Service encounter an internal error during the search, with the error being written to the user log (ULOG).
InvalidDomain	For a remote client, raised if the Bootstrap object cannot connect to the M3 domain.
InvalidName	Raised if an incorrect parameter is passed while invoking the com.beasys.Tobj_Bootstrap.resolve_initial_references method.
NoComponent	The library name component is undefined.
NotImplemented	This exception is raised if a client application invokes a method on an object that has not been implemented.
NotSet	The identifier or kind attribute of the library name component is not set.
OverFlow	The library cannot allocate resources for the inserted components.
RegistrarNotAvailable	This exception is raised if the FactoryFinder object cannot locate the CORBAservices Naming Service object.
RMfailed	The corresponding call to the tx_open() or tx_close() function failed.

Hierarchy For Package com.beasys.Tobj

Package Hierarchies:
All Packages

Class Hierarchy

- class java.lang.Object
  - class com.beasys.Tobj.**AuthType** (implements org.omg.CORBA.portable.IDLEntity)
  - class com.beasys.Tobj.**LName**
  - class com.beasys.Tobj.**LNameComponent**
  - class com.beasys.Tobj.**Server** (implements java.io.Serializable)
  - class java.lang.Throwable (implements java.io.Serializable)
    - class java.lang.Exception
      - class com.beasys.Tobj.**NoComponent**
      - class com.beasys.Tobj.**NotImplemented**
      - class com.beasys.Tobj.**NotSet**
      - class com.beasys.Tobj.**OverFlow**
      - class org.omg.CORBA.UserException (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.Tobj.**CannotProceed** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.Tobj.**InvalidDomain** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.Tobj.**InvalidName** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.Tobj.**RegistrarNotAvailable** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.Tobj.**RMfailed** (implements org.omg.CORBA.portable.IDLEntity)
  - class com.beasys.Tobj.**TP**

Interface Hierarchy

- interface org.omg.CORBA.Object
  - interface org.omg.CosTransactions.**Current**(also extends org.omg.CORBA.portable.IDLEntity)
    - interface com.beasys.Tobj.**TransactionCurrent**(also extends org.omg.CORBA.portable.IDLEntity, org.omg.CORBA.Object)
  - interface com.beasys.Tobj.**FactoryFinder**(also extends org.omg.CosLifeCycle.FactoryFinder, org.omg.CORBA.portable.IDLEntity)
  - interface org.omg.CosLifeCycle.**FactoryFinder**(also extends org.omg.CORBA.portable.IDLEntity)

        - interface com.beasys.Tobj.**FactoryFinder**(also extends org.omg.CORBA.portable.IDLEntity, org.omg.CORBA.Object)
    - interface com.beasys.Tobj.**PrincipalAuthenticator**(also extends org.omg.CORBA.portable.IDLEntity, org.omg.SecurityLevel2.PrincipalAuthenticator)
    - interface org.omg.SecurityLevel2.**PrincipalAuthenticator**(also extends org.omg.CORBA.portable.IDLEntity)
        - interface com.beasys.Tobj.**PrincipalAuthenticator**(also extends org.omg.CORBA.portable.IDLEntity, org.omg.CORBA.Object)
    - interface com.beasys.Tobj.**TransactionCurrent**(also extends org.omg.CosTransactions.Current, org.omg.CORBA.portable.IDLEntity)
- interface java.io.Serializable
    - interface org.omg.CORBA.portable.IDLEntity
        - interface org.omg.CosTransactions.**Current**(also extends org.omg.CORBA.Object)
            - interface com.beasys.Tobj.**TransactionCurrent**(also extends org.omg.CORBA.portable.IDLEntity, org.omg.CORBA.Object)
        - interface com.beasys.Tobj.**FactoryFinder**(also extends org.omg.CosLifeCycle.FactoryFinder, org.omg.CORBA.Object)
        - interface org.omg.CosLifeCycle.**FactoryFinder**(also extends org.omg.CORBA.Object)
            - interface com.beasys.Tobj.**FactoryFinder**(also extends org.omg.CORBA.portable.IDLEntity, org.omg.CORBA.Object)
        - interface com.beasys.Tobj.**PrincipalAuthenticator**(also extends org.omg.CORBA.Object, org.omg.SecurityLevel2.PrincipalAuthenticator)
        - interface org.omg.SecurityLevel2.**PrincipalAuthenticator**(also extends org.omg.CORBA.Object)
            - interface com.beasys.Tobj.**PrincipalAuthenticator**(also extends org.omg.CORBA.portable.IDLEntity, org.omg.CORBA.Object)
        - interface com.beasys.Tobj.**TransactionCurrent**(also extends org.omg.CosTransactions.Current, org.omg.CORBA.Object)

---

com.beasys.Tobj
Interface FactoryFinder

public abstract interface **FactoryFinder**
extends org.omg.CORBA.Object, org.omg.CORBA.portable.IDLEntity, FactoryFinder

The FactoryFinder interface provides clients with one object reference that serves as the single point of entry into the M3 domain. The CORBAservices Naming Service provides the mapping of factory names to object references for the FactoryFinder. Multiple FactoryFinders and the CORBAservices Naming Service together provide increased availability and reliability over the single-implementation approach used in the previous release of this product.

Fields inherited from class java.io.Serializable
`serialVersionUID`

Fields inherited from class java.io.Serializable
`serialVersionUID`

Method Summary	
`org.omg.CORBA.Object[]`	**`find_factories_by_id`**`(java.lang.String factory_id)` Obtains a list of application factories.
`org.omg.CORBA.Object`	**`find_one_factory_by_id`**`(java.lang.String factory_id)` This member function instructs the FactoryFinder to return one application factory object reference whose id in the key matches the method's input factory_id.
`org.omg.CORBA.Object`	**`find_one_factory`**`(org.omg.CosNaming.NameComponent[] factory_key)` Obtains a single application factory.
`FactoryComponent[]`	**`list_factories`**`()` Lists all of the application factory names and object references.

Methods inherited from interface org.omg.CosLifeCycle.FactoryFinder
`find_factories`

Method Detail

find_one_factory

```
public org.omg.CORBA.Object find_one_factory(org.omg.CosNaming.NameComponent[] factory_key)
                                    throws org.omg.CosLifeCycle.NoFactory,
                                           CannotProceed,
                                           RegistrarNotAvailable
```

Obtains a single application factory. This member function instructs the FactoryFinder to return one application factory object reference whose key matches the input factory_key. To accomplish this, the member function performs an equality match; that is, every NameComponent pair in the input factory_key must exactly match each pair in the application factory's key. If multiple factory keys contain the input factory_key, the FactoryFinder selects one factory key, based on an internally defined load balancing scheme. Invoking find_one_factory multiple times using the same id may return different object references.

Parameters:

`factory_key` - This parameter contains a sequence of NameComponents (<id, kind> value pairs) that uniquely identifies a factory object reference.

Returns:

Returns an object reference for an application factory. The C++ member function returns CosLifeCycle.Factory, and the Java method returns org.omg.CORBA.Object.

Throws:

NoFactory - This exception is raised if the FactoryFinder cannot find an application factory object reference that corresponds to the input factory_key.

CannotProceed - This exception is raised if the FactoryFinder or CORBAservices Naming Service encounter an internal error during the search, with the error being written to the user log (ULOG). Notify the operations staff immediately if this exception is raised. Depending on the severity of the internal error, the server running the FactoryFinder or CORBAservices Naming Service may have terminated. If a FactoryFinder service has terminated, start a new FactoryFinder service. If a CORBAservices Naming Service has terminated and there is another CORBAservices Naming Service running, start a new CORBAservices Naming Service. If no naming services servers are running, restart the application.

RegistrarNotAvailable - This exception is raised if the FactoryFinder object cannot locate the CORBAservices Naming Service object. Notify the operations staff immediately if this exception is raised. If no naming services servers are running, restart the application.

---

find_one_factory_by_id

```
public org.omg.CORBA.Object find_one_factory_by_id(java.lang.String factory_id)
                                          throws org.omg.CosLifeCycle.NoFactory,
                                                 CannotProceed,
                                                 RegistrarNotAvailable
```

This member function instructs the FactoryFinder to return one application factory object reference whose id in the key matches the method's input factory_id. To accomplish this, the member function performs an equality match (that is, the input factory_id must exactly match the id in the pair in the application factory's key). If multiple factory keys contain the input factory_id, the FactoryFinder selects one factory key, based on an internally defined load

balancing scheme. Invoking the find_one_factory_by_id method multiple times using the same id may return different object references.

Parameters:

`factory_id` - This parameter represents a string identifier that is used to identify the id or type of application factory. For some suggestions as to the composition of this string, see Creating Java Server Applications.

Returns:

Returns an object reference for an application factory. The C++ member function returns a CosLifeCycle.Factory, and the Java method returns org.omg.CORBA.Object.

Throws:

NoFactory - This exception is raised if the FactoryFinder cannot find an application factory object reference that corresponds to the input factory_id.

CannotProceed - This exception is raised if the FactoryFinder or CORBAservices Naming Service encounter an internal error during the search, with the error being written to the user log (ULOG). Notify the operations staff immediately if this exception is raised. Depending on the severity of the internal error, the server running the FactoryFinder or the CORBAservices Naming Service may have terminated. If a FactoryFinder service has terminated, start a new FactoryFinder service. If a CORBAservices Naming Service has terminated and there is another CORBAservices Naming Service running, start a new CORBAservices Naming Service. If no naming services servers are running, restart the application.

RegistrarNotAvailable - This exception is raised if the FactoryFinder object cannot locate the CORBAservices Naming Service object. Notify the operations staff immediately if this exception is raised. If no naming service servers are running, restart the application.

---

find_factories_by_id

```
public org.omg.CORBA.Object[] find_factories_by_id(java.lang.String factory_id)
                                          throws org.omg.CosLifeCycle.NoFactory,
                                                 CannotProceed,
                                                 RegistrarNotAvailable
```

Obtains a list of application factories. This member function instructs the FactoryFinder to return a list of application factory object references whose id in the keys match the method's input factory_id. To accomplish this, the member function performs an equality match (that is, the input factory_id must exactly match each id in the pair in the application factory's keys).

Parameters:

`factory_id` - This parameter represents a string identifier that is used to identify the kind or type of application factory. For some suggestions as to the composition of this string, see Creating Client Applications.

Returns:

Returns a sequence of object references for application factories. The C++ member function returns CosLifeCycle.Factories, and the Java method returns org.omg.CORBA.Object[].

Throws:

NoFactory - This exception is raised if the FactoryFinder cannot find an application factory object reference that corresponds to the input factory_key or factory_id.

CannotProceed - This exception is raised if the FactoryFinder or CORBAservices Naming Service encounter an internal error during the search with the error being written to the user log (ULOG). Notify the operations staff immediately if this exception is raised. Depending on the severity of the internal error, the server running the FactoryFinder or CORBAservices

Naming Service may have terminated. If a FactoryFinder service has terminated, start a new FactoryFinder service. If a CORBAservices Naming Service has terminated and there is another CORBAservices Naming Service running, start a new CORBAservices Naming Service. If no naming services servers are running, restart the application.

RegistrarNotAvailable - This exception is raised if the FactoryFinder object cannot locate the CORBAservices Naming Service object. Notify the operations staff immediately if this exception is raised. If no naming service servers are running, restart the application.

---

list_factories

```
public FactoryComponent[] list_factories()
                                  throws CannotProceed,
                                         RegistrarNotAvailable
```

Lists all of the application factory names and object references. This method instructs the FactoryFinder to return a list containing all of the factory keys and associated object references for application factories registered with the CORBAservices Naming Service.

Returns:

Returns Tobj.FactoryListing, which is a sequence of Tobj.FactoryComponents where each component contains the factory key that conforms to CosNaming.Name, and the corresponding object reference for the application factory.

Throws:

CannotProceed - This exception is raised if the FactoryFinder or the CORBAservices Naming Service encounter an internal error during the search, with the error being written to the user log (ULOG). Notify the operations staff immediately if this exception is raised. Depending on the severity of the internal error, the server running the FactoryFinder or CORBAservices Naming Service may have terminated. If a FactoryFinder service has terminated, start a new FactoryFinder service. If a CORBAservices Naming Service has terminated and there is another CORBAservices Naming Service running, start a new CORBAservices Naming Service. If no naming services servers are running, restart the application.

RegistrarNotAvailable - This exception is raised if the FactoryFinder object cannot locate the CORBAservices Naming Service object. Notify the operations staff immediately if this exception is raised. It is possible that no naming service servers are running. Restart the application.

---

com.beasys.Tobj
Interface PrincipalAuthenticator

public abstract interface **PrincipalAuthenticator**
extends org.omg.CORBA.Object, org.omg.CORBA.portable.IDLEntity, PrincipalAuthenticator

The PrincipalAuthenticator interface is used to log on to and log off of the M3 domain.

Fields inherited from class java.io.Serializable
`serialVersionUID`

Fields inherited from class java.io.Serializable
`serialVersionUID`

Method Summary

`void`	**`build_auth_data`**`(java.lang.String user_name, java.lang.String client_name, java.lang.String system_password, java.lang.String user_password, byte[] user_data, OpaqueHolder auth_data, AttributeListHolder privileges)` Creates authentication data and attributes for use by the SecurityLevel2.PrincipalAuthenticator.authenticate method.
`AuthType`	**`get_auth_type`**`()` Gets the type of authentication expected by the M3 domain.
`void`	**`logoff`**`()` Discards the M3 client authentication context, but does not close the network connections to the M3 domain.
`AuthenticationStatus`	**`logon`**`(java.lang.String user_name, java.lang.String client_name, java.lang.String system_password, java.lang.String user_password, byte[] user_data)` Authenticates the client.

Methods inherited from interface org.omg.SecurityLevel2.PrincipalAuthenticator
authenticate, continue_authentication

Method Detail

get_auth_type

```
public AuthType get_auth_type()
```

Gets the type of authentication expected by the M3 domain.

Note: This method raises the org.omg.CORBA.BAD_INV_ORDER exception if it is called with an invalid SecurityCurrent object.

Returns:

Returns the type of authentication required to access the M3 domain.

---

logon

```
public AuthenticationStatus logon(java.lang.String user_name,
                                  java.lang.String client_name,
                                  java.lang.String system_password,
                                  java.lang.String user_password,
                                  byte[] user_data)
```

Authenticates the client. For remote M3 clients, this method authenticates the client via the IIOP Server Listener/Handler so that the remote client can access an M3 domain. This method is functionally equivalent to the org.omg.SecurityLevel2.PrincipalAuthenticator.authenticate method, but the parameters are oriented to M3 security.

Note: This method raises the org.omg.CORBA.BAD_INV_ORDER exception if it is called with an invalid SecurityCurrent object.

Parameters:

`user_name` - The M3 user name. The authentication level is com.beasys.Tobj.AuthType.TOBJ_NOAUTH. If user_name is null or empty, or exceeds 30 characters, logon raises the org.omg.CORBA.BAD_PARAM exception.

`client_name` - The M3 client name. The authentication level is TOBJ_NOAUTH. If the client_name is NULL or empty, or exceeds 30 characters, logon raises the org.omg.CORBA.BAD_PARAM exception.

`system_password` - The M3 client application password. The authentication level is com.beasys.Tobj.AuthType.TOBJ_SYSAUTH. If the client name is NULL or empty, logon raises the org.omg.CORBA.BAD_PARAM exception.

Note: The system_password must not exceed eight characters.

`user_password` - The user password (needed for use by the default M3 authentication service). The authentication level is com.beasys.Tobj.AuthType.TOBJ_APPAUTH.

`user_data` - Data that is specific to the client application (needed for use by a custom M3 authentication service). The authentication level is com.beasys.Tobj.AuthType.TOBJ_APPAUTH.

Note: The TOBJ_SYSAUTH authentication type includes the requirements of the TOBJ_NOAUTH type, plus a client application password. The TOBJ_APPAUTH authentication type includes the requirements of the TOBJ_SYSAUTH authentication type, plus additional information, such as a user password or user data.

Note: The user_password and user_data parameters are mutually exclusive, depending on the requirements of the authentication service used in the configuration of the M3 domain. The M3 default authentication service expects a user password. A customized authentication service may require user data. The logon call raises the org.omg.CORBA.BAD_PARAM exception if both user_password and user_data are specified.

Returns:

org.omg.Security.AuthenticationStatus.SecAuthSuccess if the authentication succeeded.

org.omg.Security.AuthenticationStatus.SecAuthFailure if the authentication failed, or if the client was already authenticated and did not invoke the com.beasys.Tobj.PrincipalAuthenticator.logoff or com.beasys.Tobj_Bootstrap.destroy_current methods.

---

logoff

```
public void logoff()
```

Discards the M3 client authentication context, but does not close the network connections to the M3 domain. Logoff also invalidates the current credentials. After logging off, invocations using existing object references fail if the authentication type is not com.beasys.Tobj.AuthType.TOBJ_NOAUTH.

If the client is currently authenticated to an M3 domain, invoking the com.beasys.Tobj_Bootstrap.destroy_current method invokes logoff implicitly.

Note: This method raises the org.omg.CORBA.BAD_INV_ORDER exception if it is called with an invalid SecurityCurrent object.

---

build_auth_data

```
public void build_auth_data(java.lang.String user_name,
                            java.lang.String client_name,
                            java.lang.String system_password,
                            java.lang.String user_password,
                            byte[] user_data,
                            OpaqueHolder auth_data,
                            AttributeListHolder privileges)
```

Creates authentication data and attributes for use by the SecurityLevel2.PrincipalAuthenticator.authenticate method.

Note: This method raises the org.omg.CORBA.BAD_INV_ORDER exception if it is called with an invalid SecurityCurrent object.

Parameters:

`user_name` - The M3 user name.

`client_name` - The M3 client name.

`system_password` - The M3 client application password.

`user_password` - The user password (default M3 authentication service).

`user_data` - Client application-specific data (custom M3 authentication service).

`auth_data` - For use by authenticate.

`privileges` - For use by authenticate.

Note: If user_name or client_name is NULL or empty, or exceeds 30 characters, the subsequent authenticate method invocation raises the org.omg.CORBA.BAD_PARAM exception.

Note: The user_password and user_data parameters are mutually exclusive, depending on the requirements of the authentication service used in the configuration of the M3 domain. The M3 default authentication service expects a user password. A customized authentication service may require user data. If both user_password and user_data are specified, the subsequent authentication call raises the org.omg.CORBA.BAD_PARAM exception.

---

com.beasys.Tobj
Interface TransactionCurrent

public abstract interface **TransactionCurrent**
extends org.omg.CORBA.Object, org.omg.CORBA.portable.IDLEntity, Current

The TransactionCurrent interface supports all the methods of the Current interface in the CosTransactions module, as described in the previous section. In addition, this interface supports APIs to open and close the resource manager.

Fields inherited from class java.io.Serializable
`serialVersionUID`

Fields inherited from class java.io.Serializable
`serialVersionUID`

Method Summary

`void`	**`close_xa_rm()`** This method closes the XA resource manager to which this process is linked.
`void`	**`open_xa_rm()`** This method opens the XA resource manager to which this process is linked.

Methods inherited from interface org.omg.CosTransactions.Current
`begin, commit, get_control, get_status, get_transaction_name, resume, rollback_only, rollback, set_timeout, suspend`

Method Detail

open_xa_rm

```
public void open_xa_rm()
                throws RMfailed
```

This method opens the XA resource manager to which this process is linked. XA resource managers are provided by database vendors, such as Oracle and Informix.

The open_xa_rm method should be invoked in place of an open invocation that is specific to a resource manager. Because resource managers differ in their initialization semantics, the specific information needed to open a particular resource manager is placed in the OPENINFO parameter in the GROUPS section of the UBBCONFIG file. The format of the OPENINFO string is dependent on the requirements of the database vendor providing the underlying resource manager. For more information about the OPENINFO parameter, see the ubbconfig(5) reference page in the BEA TUXEDO Reference and the Administration Guide. Also, refer to database vendor documentation for information about how to develop and install applications that use the XA libraries.

Any attempts to invoke this method by a client will raise an org.omg.CORBA.NO_IMPLEMENT standard system exception.

Throws:

RMfailed - Is raised if there is a failure while opening the resource manager.

---

close_xa_rm

```
public void close_xa_rm()
                 throws RMfailed
```

This method closes the XA resource manager to which this process is linked. XA resource managers are provided by database vendors, such as Oracle and Informix.

The close_xa_rm() method should be invoked in place of a close invocation that is specific to the resource manager. Because resource managers differ in their termination semantics, the specific information needed to close a particular resource manager is placed in the CLOSEINFO parameter in the GROUPS section of the M3 system UBBCONFIG file.

The format of the CLOSEINFO string is dependent on the requirements of the database vendor providing the underlying resource manager. For more information about the CLOSEINFO parameter, see the ubbconfig(5) reference page in the BEA TUXEDO Reference online document and the Administration Guide. Also, refer to database vendor documentation for information about how to develop and install applications that use the XA libraries.

An org.omg.CORBA.BAD_INV_ORDER standard system exception is raised if the function was called in an improper context (for example, the caller is in transaction mode).

Any attempts to invoke this method by the lightweight clients or the native clients will raise an org.omg.CORBA.NO_IMPLEMENT standard system exception.

Throws:

RMfailed - Is raised if there is a failure while closing the resource manager.

com.beasys.Tobj
Class AuthType

```
java.lang.Object
  |
  +--com.beasys.Tobj.AuthType
```

public final class **AuthType**
extends java.lang.Object
implements org.omg.CORBA.portable.IDLEntity

The type of authentication expected by the M3 domain.

See Also:
Serialized Form

Field Summary

`static AuthType`	**`TOBJ_APPAUTH`** The client must provide information in addition to that which is required by TOBJ_SYSAUTH.
`static AuthType`	**`TOBJ_NOAUTH`** No authentication is needed; however, the client can still authenticate itself, and must specify a user name and a client name, but no password.
`static AuthType`	**`TOBJ_SYSAUTH`** The client must authenticate itself to the M3 domain, and must specify a user name, client name, and client application password.

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

TOBJ_NOAUTH

```
public static final AuthType TOBJ_NOAUTH
```

No authentication is needed; however, the client can still authenticate itself, and must specify a user name and a client name, but no password.

TOBJ_SYSAUTH

```
public static final AuthType TOBJ_SYSAUTH
```

The client must authenticate itself to the M3 domain, and must specify a user name, client name, and client application password.

TOBJ_APPAUTH

```
public static final AuthType TOBJ_APPAUTH
```

The client must provide information in addition to that which is required by TOBJ_SYSAUTH. If the default M3 authentication service is used in the M3 domain configuration, the client must provide a user password; otherwise, the client provides authentication data that is interpreted by the custom authentication service in the M3 domain.

---

com.beasys.Tobj
Class LName

```
java.lang.Object
  |
  +--com.beasys.Tobj.LName
```

public class **LName**
extends java.lang.Object

This class implements the names library in CosNaming. This class is a Java-specific implementation of an OMG-defined CORBA pseudo-object, which will be deprecated in a future release.

Method Summary	
`static LName`	**`create_lname`**`()` Creates a library name pseudo-object.
`LNameComponent`	**`delete_component`**`(long long_i)` Removes and returns the component specified by `long_i`.
`void`	**`destroy`**`()` Destroys library name component pseudo-objects.
`boolean`	**`equal`**`(LName lnp)` Tests for equality with the library name specified by `ln`.
`void`	**`from_idl_form`**`(org.omg.CosNaming.NameComponent[] nr)` Sets the components and kind attribute for a library name from a returned IDL defined structure represented by `nr`.
`LNameComponent`	**`get_component`**`(long long_i)` Returns the component specified by `long_i`.
`LName`	**`insert_component`**`(long long_i, LNameComponent ncp)` Inserts a component after the position specified by `long_i`.
`boolean`	**`less_than`**`(LName lnp)` Tests for the order of a library name in relation to the library name specified by `lnp`.
`long`	**`num_components`**`()` Returns the number of components in a library name.
`org.omg.CosNaming.NameComponent[]`	**`to_idl_form`**`()` Produces a structure that can be passed across the IDL request.

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Method Detail

create_lname

```
public static LName create_lname()
```

Creates a library name pseudo-object.

---

destroy

```
public void destroy()
```

Destroys library name component pseudo-objects.

---

insert_component

```
public LName insert_component(long long_i,
                              LNameComponent ncp)
                       throws NoComponent,
                              OverFlow
```

Inserts a component after the position specified by `long_i`.

---

get_component

```
public LNameComponent get_component(long long_i)
                             throws NoComponent
```

Returns the component specified by `long_i`.

---

delete_component

```
public LNameComponent delete_component(long long_i)
                                throws NoComponent
```

Removes and returns the component specified by `long_i`.

---

num_components

```
public long num_components()
```

Returns the number of components in a library name.

---

equal

```
public boolean equal(LName lnp)
```

Tests for equality with the library name specified by `ln`.

---

less_than

```
public boolean less_than(LName lnp)
                  throws NotImplemented
```

Tests for the order of a library name in relation to the library name specified by `lnp`.

---

from_idl_form

```
public void from_idl_form(org.omg.CosNaming.NameComponent[] nr)
                   throws OverFlow
```

Sets the components and kind attribute for a library name from a returned IDL defined structure represented by `nr`.

---

to_idl_form

```
public org.omg.CosNaming.NameComponent[] to_idl_form()
                                                 throws InvalidName
```

Produces a structure that can be passed across the IDL request.

---

com.beasys.Tobj
Class LNameComponent

```
java.lang.Object
  |
  +--com.beasys.Tobj.LNameComponent
```

public class **LNameComponent**
extends java.lang.Object

This class implements the names library in CosNaming. This class is a Java-specific implementation of an OMG-defined CORBA pseudo-object, which will be deprecated in a future release.

Method Summary

`static LNameComponent`	**`create_lname_component`**`()` Creates the LName component.
`java.lang.String`	**`get_id`**`()` Returns the value of the LName component's identifier attribute.
`java.lang.String`	**`get_kind`**`()` Returns the value of the LName component's kind attribute.
`void`	**`set_id`**`(java.lang.String i)` Sets the identifier attribute of the LName component to the string argument.
`void`	**`set_kind`**`(java.lang.String k)` Sets the kind attribute of the LName component to the string argument.

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Method Detail

create_lname_component

```
public static LNameComponent create_lname_component()
```

Creates the LName component.

---

get_id

```
public java.lang.String get_id()
                        throws NotSet
```

Returns the value of the LName component's identifier attribute.

---

set_id

```
public void set_id(java.lang.String i)
            throws OverFlow
```

Sets the identifier attribute of the LName component to the string argument.

---

get_kind

```
public java.lang.String get_kind()
                          throws NotSet
```

Returns the value of the LName component's kind attribute.

---

set_kind

```
public void set_kind(java.lang.String k)
              throws OverFlow
```

Sets the kind attribute of the LName component to the string argument.

---

com.beasys.Tobj
Class Server

```
java.lang.Object
  |
  +--com.beasys.Tobj.Server
```

public class **Server**
extends java.lang.Object
implements java.io.Serializable

Provides callback methods that can be used for application-specific server initialization and termination logic. Default implementations are provided.

See Also:
Serialized Form

Constructor Summary

Server()	

Method Summary

boolean	**initialize**(java.lang.String[] args) Allows the application to perform application-specific initialization procedures, such as logging into a database, creating and registering well-known object factories, initializing global variables, and so forth.
void	**release**() Allows the application to perform any application-specific cleanup, such as logging off a database, unregistering well-known factories, or deallocating resources.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Server

```
public Server()
```

Method Detail

initialize

```
public boolean initialize(java.lang.String[] args)
                  throws InitializeFailed
```

Allows the application to perform application-specific initialization procedures, such as logging into a database, creating and registering well-known object factories, initializing global variables, and so forth.

Note: Default implementation opens default XA RM.

The com.beasys.Tobj.Servant.initialize method, which is invoked as the last step in server initialization, allows the application to perform application-specific initialization. Typically, a server application does the following tasks in the initialize method:

- Creates references for CORBA object factories implemented in the server application and registers them with the FactoryFinder using the com.beasys.Tobj.TP.register_factory method.
- Initializes global variables, if any are used.
- Opens XA resource managers if any are used by the server application.

The TP Framework provides a default implementation of the initialize method. The default implementation opens XA resource managers for the server. The TP Framework does this by issuing a tx_open() invocation, which uses the default OPENINFO parameter configured for the server's group in the UBBCONFIG file or the TUXCONFIG file.

Note: If the server application implements the initialize method, it is the responsibility of the server application to open any required XA resource managers. This is done by invoking either of the following methods:

- com.beasys.Tobj.TP.open_xa_rm
  This is the preferred technique for server applications, since it can be done on a static function, without the need to obtain an object reference.
- com.beasys.Tobj.TransactionCurrent.open_xa_rm
  A reference to the TransactionCurrent object can be obtained from the Bootstrap object.

  Note: This behavior of initialize method is different from previous releases of the M3 software. In earlier versions the server application had to call com.beasys.Tobj.TP.open_xa_rm() explicitly, even for server applications using the null XA interface. For this release of the M3 software, the call can be done implicitly by not

implementing the initialize method.

Transactions may be started in the initialize method after invoking the com.beasys.Tobj.TransactionCurrent.open_xa_rm or com.beasys.Tobj.TP.open_xa_rm method. However, any transactions that are started in the initialize method must be terminated by the server application before the initialize method returns. If the transactions are still active when control is returned, the server system fails to boot, and it exits gracefully. This happens because the server application has no logical way of either committing or rolling back the transaction after the initialize method returns.

Parameters:

`args` - Command line options. The argv[0] argument contains the name of the server.

Returns:

TRUE indicates success. FALSE indicates failure. If an error occurs in initialize(), the application code should return FALSE. The application code should not call the system call exit(). Invoking exit() does not give the TP Framework a chance to release resources allocated during startup and may cause unpredictable results.

If the return value is FALSE:

- All resources allocated by the server will be released and the server will be shut down.
- The com.beasys.Tobj.Server.release method is not invoked.
- Any transactions that are started in the initialize method and are not terminated will eventually time out; they are not automatically rolled back.

Throws:

InitializeFailed - If an exception is raised in the initialize method, the TP Framework catches the exception. The TP Framework behavior is the same as if the initialize method returned FALSE (that is, an exception is considered to be a failure). In addition, an error message is written to the user log (ULOG) file.

---

release

```
public void release()
            throws ReleaseFailed
```

Allows the application to perform any application-specific cleanup, such as logging off a database, unregistering well-known factories, or deallocating resources.

Note: Default implementation closes default XA RM.

Throws:

ReleaseFailed - A ULOG message is reported, and server process shutdown continues.

---

com.beasys.Tobj

Class TP

```
java.lang.Object
  |
  +--com.beasys.Tobj.TP
```

public class **TP**
extends java.lang.Object

Supplies a set of service methods that can be invoked by application code. This is the only interface in the TP Framework that can safely be invoked by application code. All other interfaces have callback methods that are intended to be invoked only by system code.

Constructor Summary

TP()	

Method Summary	
static Tobj_Bootstrap	**bootstrap**() Returns a com.beasys.Tobj_Bootstrap object.
static void	**close_xa_rm**() Closes the XA resource manager to which the invoking process is linked.
static org.omg.CORBA.Object	**create_object_reference**(java.lang.String interfaceName, java.lang.String stroid, org.omg.CORBA.NVList criteria) Creates an object reference.
static void	**deactivateEnable**() Enables application-controlled deactivation of CORBA objects.
static org.omg.CORBA.Object	**get_object_reference**() Returns a pointer to the current object.
static int	**open_xa_rm**() Opens the XA resource manager to which the invoking process is linked.
static org.omg.CORBA.ORB	**orb**() Returns an ORB object.
static void	**register_factory**(org.omg.CORBA.Object objp, java.lang.String id) Locates the M3 FactoryFinder object and registers an M3 factory.
static void	**unregister_factory**(org.omg.CORBA.Object objp, java.lang.String id) Locates the M3 FactoryFinder object and removes an M3 factory.
static int	**userlog**(java.lang.String str) Writes a message to the user log (ULOG) file.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

TP

public **TP**()

Method Detail

create_object_reference

```
public static org.omg.CORBA.Object create_object_reference(java.lang.String interfaceName,
                                                           java.lang.String stroid,
                                                           org.omg.CORBA.NVList criteria)
                                                    throws InvalidInterface,
                                                           InvalidObjectId
```

Creates an object reference. The resulting object reference may be passed to clients who use it to access the object. The server application is responsible for invoking the create_object_reference method.

Ordinarily, the server application calls this method in two places:

- In the com.beasys.Tobj.Server.initialize method to create factories for the server.
- In factory methods to create object references to be returned to clients.

For examples of how and when to call the create_object_reference method, see Creating Java Server Applications.

Parameters:

`interfaceName` - Specifies a character string that contains the fully qualified interface name for the object.

`stroid` - Specifies the object ID in string format. The object ID uniquely identifies this instance of the class. It is up to the programmer to decide what information to place in the object ID. One possibility would be to use the object ID to hold a database key. Choosing the value of an object identifier, and the degree of uniqueness, is part of the application design. The M3 software cannot guarantee any uniqueness in object references, since object references may be legitimately copied and shared outside the M3 domain (for example, by passing the object reference as a string).

`criteria` - Specifies a list of named values that can be used to provide factory-based routing for the object reference. The use of factory-based routing is optional and is dependent on the use of this argument. If you do not want to use factory-based routing, you can pass a value of 0 (zero) for this argument.

Returns:

The newly created object reference.

Throws:

InvalidInterface - Indicates that the specified interface_name is null.
InvalidObjectId - Indicates that the specified stroid is null.

---

get_object_reference

```
public static org.omg.CORBA.Object get_object_reference()
                                                 throws NilObject
```

Returns a pointer to the current object.

Returns:

The current object when invoked within the scope of a CORBA object execution. Otherwise, the com.beasys.TobjS.NilObject exception is raised.

Note: If the get_object_reference method is invoked from within either the com.beasys.com.Tobj.Server.initialize or com.beasys.com.Tobj.Server.release methods, it is considered to be invoked outside the scope of an application's CORBA object execution.

Throws:

NilObject - Indicates that the method was invoked outside the scope of an application's CORBA object execution. The reason string contains OutOfScope.

---

register_factory

```
public static void register_factory(org.omg.CORBA.Object objp,
                                    java.lang.String id)
                             throws InvalidObject,
                                    InvalidName,
                                    RegistrarNotAvailable,
                                    CannotProceed,
                                    OverFlow
```

Locates the M3 FactoryFinder object and registers an M3 factory. Typically, the register_factory method is invoked from the com.beasys.Tobj.Server.initialize method when the server creates its factories.

Parameters:

`objp` - Specifies the object reference that was created for an application factory using the com.beasys.Tobj.TP.create_object_reference method.

`id` - Specifies a string identifier that is used to identify the application factory. For some suggestions as to the composition of this string, see Creating Java Server Applications.

Throws:

InvalidObject - Indicates that the factory value is null.

InvalidName - Indicates that the id string is empty. It is also raised if the field contains blank spaces or control characters.

RegistrarNotAvailable - Indicates that the FactoryFinder Registrar object cannot locate the CORBAservices Naming Service object. Notify the operations staff immediately if this exception is raised. If no naming services servers are running, restart the application.

CannotProceed - Indicates that the FactoryFinder Registrar object encountered an internal error during the registration process. The operations staff should be notified immediately if this exception is raised. A possible situation is that the Registrar object's internal tables have been corrupted. The FactoryFinder server should be restarted using the backup file.

OverFlow - Indicates that the id string is longer than 128 bytes (currently the maximum allowable length).

---

unregister_factory

```
public static void unregister_factory(org.omg.CORBA.Object objp,
                                      java.lang.String id)
                               throws InvalidObject,
```

```
                    InvalidName,
                    RegistrarNotAvailable,
                    CannotProceed,
                    OverFlow
```

Locates the M3 FactoryFinder object and removes an M3 factory. Typically, the unregister_factory method is invoked from the com.beasys.Tobj.Server.release method to unregister server factories.

Parameters:

`objp` - Specifies the object reference that was created for an application factory using the com.beasys.Tobj.TP.create_object_reference method.

`id` - Specifies a string identifier that is used to identify the application factory. For some suggestions as to the composition of this string, see Creating Java Server Applications.

Throws:

InvalidObject - Indicates that the factory value is null.

InvalidName - Indicates that the id string is empty. It is also raised if the field contains blank spaces or control characters.

RegistrarNotAvailable - Indicates that the FactoryFinder Registrar object cannot locate the CORBAservices Naming Service object. Notify the operations staff immediately if this exception is raised. If no naming services servers are running, restart the application.

CannotProceed - Indicates that the FactoryFinder Registrar object encountered an internal error during the registration process. The operations staff should be notified immediately if this exception is raised. A possible situation is that the Registrar object's internal tables have been corrupted. The FactoryFinder server should be restarted using the backup file.

OverFlow - Indicates that the id string is longer than 128 bytes (currently the maximum allowable length).

---

open_xa_rm

```
public static int open_xa_rm()
                    throws RMfailed
```

Opens the XA resource manager to which the invoking process is linked. XA resource managers are provided by database vendors, such as Oracle and Informix.

Note: The functionality of this method is also provided by the com.beasys.Tobj.TransactionCurrent.open_xa_rm method. However, the open_xa_rm method provides a more convenient way for a server application to open a resource manager because there is no need to obtain an object reference to the TransactionCurrent object. A reference to the TransactionCurrent object can be obtained from the Bootstrap object.

This method should be invoked once from the com.beasys.Tobj.Server.initialize method for each server that participates in a global transaction. This includes servers that are linked with an XA resource manager, as well as servers that participate in a global transaction, but are not actually linked with an XA-compliant resource manager.

The open_xa_rm method should be invoked in place of an open invocation that is specific to a resource manager. Because resource managers differ in their initialization semantics, the specific information needed to open a particular resource manager is placed in the OPENINFO parameter in the GROUPS section of the UBBCONFIG file.

The format of the OPENINFO string is dependent on the requirements of the database vendor providing the underlying resource manager. For more information about the OPENINFO parameter, see the ubbconfig(5) reference page in the BEA TUXEDO Reference and the Administration Guide. Also, refer to database vendor documentation for information about how to develop and install applications that use the XA libraries.

Note: Only one resource manager can be linked to the invoking process.

Returns:
Upon successful completion, the open_xa_rm method returns 0 (zero).

Throws:
RMfailed - The tx_open() call failed.

---

close_xa_rm

```
public static void close_xa_rm()
                        throws RMfailed,
                               org.omg.CORBA.BAD_INV_ORDER
```

Closes the XA resource manager to which the invoking process is linked. XA resource managers are provided by database vendors, such as Oracle and Informix.

Note: The functionality of this call is also provided by the com.beasys.Tobj.TransactionCurrent.close_xa_rm method. The com.beasys.Tobj.TP.close_xa_rm method provides a more convenient way for a server application to close a resource manager because there is no need to obtain an object reference to the TransactionCurrent object. A reference to the TransactionCurrent object can be obtained from the Bootstrap object.

This method should be invoked once from the com.beasys.Tobj.Server.release method for each server that is involved in global transactions. This includes servers that are linked with an XA resource manager, as well as servers that are involved in global transactions, but are not actually linked with an XA-compliant resource manager.

The com.beasys.Tobj.close_xa_rm method should be invoked in place of a close invocation that is specific to the resource manager. Because resource managers differ in their termination semantics, the specific information needed to close a particular resource manager is placed in the CLOSEINFO parameter in the GROUPS section of the M3 system UBBCONFIG file.

The format of the CLOSEINFO string is dependent on the requirements of the database vendor providing the underlying resource manager. For more information about the CLOSEINFO parameter, see the ubbconfig(5) reference page in the BEA TUXEDO Reference online document and the Administration Guide. Also, refer to database vendor documentation for information about how to develop and install applications that use the XA libraries.

Throws:
RMfailed - The tx_open() call failed.

org.omg.CORBA.BAD_INV_ORDER - There is an active transaction. The resource manager cannot be closed while a transaction is active.

---

deactivateEnable

```
public static void deactivateEnable()
                            throws IllegalOperation
```

Enables application-controlled deactivation of CORBA objects. This method can be used to cause deactivation of an object upon completion of the method in which it is invoked. The deactivateEnable method enables application-controlled deactivation of CORBA objects. It provides greater flexibility than can be obtained by the system-controlled behavior of the method, transaction, and process activation policies.

The following usage guidelines apply:

- The deactivateEnable method must be invoked within a method of a CORBA object. If you call this method from inside the com.beasys.Tobj.Server.initialize or com.beasys.Tobj.Server.release method, an appropriate exception is thrown.
- Only the object in which the deactivateEnable method is invoked is affected.
- The deactivateEnable method should be used only in conjunction with the process activation policy.

  Note: When it is invoked from within the method of an object that has its activation policy set to method, the effect is the same as the normal behavior of such objects (effectively, a NO-OP). When it is invoked within the method of an object that has its activation policy set to transaction, an IllegalOperation exception is raised. This is because deactivation of such objects may interfere with their correct notification of transaction completion by the M3 transaction manager.

  Throws:

  IllegalOperation - Indicates that the deactivateEnable method was invoked by an object with the activation policy set to transaction.

---

orb

```
public static org.omg.CORBA.ORB orb()
```

Returns an ORB object. Access to the ORB object allows the application to invoke ORB operations, such as the org.omg.CORBA.ORB.string_to_object and org.omg.CORBA.ORB.object_to_string methods. Because the ORB object is owned by the TP Framework, the invoker of the orb method should not dispose of the ORB object.

Returns:

The ORB object that is created by the TP Framework when the server application is started.

---

bootstrap

```
public static Tobj_Bootstrap bootstrap()
```

Returns a com.beasys.Tobj_Bootstrap object. The Bootstrap object is used to access initial object references for the FactoryFinder object, the Interface Repository, the TransactionCurrent, and the SecurityCurrent objects. The TP Framework creates a com.beasys.Tobj_Bootstrap object as part of initialization; it is not necessary for the application code to create any other com.beasys.Tobj_Bootstrap objects in the server.

Caution: Because the TP Framework owns the com.beasys.Tobj_Bootstrap object, server application code must not dispose of the Bootstrap object.

Returns:

The com.beasys.Tobj_Bootstrap object that is created by the TP Framework when the server application is started.

---

userlog

```
public static int userlog(java.lang.String str)
```

Writes a message to the user log (ULOG) file. Messages are appended to the ULOG file with a tag made up of the time (hhmmss), system name, process name, and process-id of the invoking process. The tag is terminated with a colon.

It is recommended that the server applications limit their use of userlog() messages to messages that can be used to help debug application errors; flooding the ULOG file with incidental information can make it difficult to spot actual errors.

Parameters:

`str` - The message to be written to the ULOG.

Returns:

The number of characters that were output, or a negative value if an output error was encountered. Output errors include the inability to open or write to the current log file.

Example:

The following example shows how to use the userlog method:

```
userlog ("System exception caught: %s", e.get_id());
```

---

com.beasys.Tobj

Class CannotProceed

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.Tobj.CannotProceed
```

public final class **CannotProceed**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

This exception is raised if the FactoryFinder or the CORBAservices Naming Service encounter an internal error during the search, with the error being written to the user log (ULOG). Notify the operations staff immediately if this exception is raised. Depending on the severity of the internal error, the server running the FactoryFinder or CORBAservices Naming Service may have terminated. If a FactoryFinder service has terminated, start a new FactoryFinder service. If a CORBAservices Naming Service has terminated and there is another CORBAservices Naming Service running, start a new CORBAservices Naming Service. If no naming services servers are running, restart the application.

See Also:
Serialized Form

Constructor Summary

CannotProceed() This is the constructor for the class.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

CannotProceed

```
public CannotProceed()
```

This is the constructor for the class.

---

com.beasys.Tobj
Class InvalidDomain

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.Tobj.InvalidDomain
```

public final class **InvalidDomain**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

For a remote client, raised if the Bootstrap object cannot connect to the M3 domain. The address of the M3 domain IIOP Server Listener/Handler is specified in the constructor. For a native client or server, raised if the domain is not booted.

See Also:
Serialized Form

Constructor Summary

`InvalidDomain``()` This is the constructor for the class.

Methods inherited from class java.lang.Throwable
`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

InvalidDomain

```
public InvalidDomain()
```

This is the constructor for the class.

---

com.beasys.Tobj
Class InvalidName

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.Tobj.InvalidName
```

public final class **InvalidName**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

Raised if an incorrect parameter is passed while invoking the com.beasys.Tobj_Bootstrap.resolve_initial_references method. On the server application, this exception is also raised when the SecurityCurrent object is passed.

See Also:
Serialized Form

Constructor Summary

`InvalidName()` This is the constructor for the class.

Methods inherited from class java.lang.Throwable
`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

InvalidName

```
public InvalidName()
```

This is the constructor for the class.

---

com.beasys.Tobj
Class NoComponent

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--com.beasys.Tobj.NoComponent
```

public class **NoComponent**
extends java.lang.Exception

The library name component is undefined. This exception is used by the LName and LNameComponent classes.

See Also:
Serialized Form

Constructor Summary

NoComponent() This is the constructor for the class.	
NoComponent(java.lang.String msg) This is the constructor for the class.	

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

NoComponent

```
public NoComponent()
```

This is the constructor for the class.

NoComponent

```
public NoComponent(java.lang.String msg)
```

This is the constructor for the class.

---

com.beasys.Tobj
Class NotImplemented

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--com.beasys.Tobj.NotImplemented
```

public class **NotImplemented**
extends java.lang.Exception

This exception is raised if a client application invokes a method on an object that has not been implemented.

See Also:
Serialized Form

Constructor Summary

NotImplemented()	

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

NotImplemented

```
public NotImplemented()
```

---

com.beasys.Tobj
Class NotSet

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--com.beasys.Tobj.NotSet
```

public class **NotSet**
extends java.lang.Exception

The identifier or kind attribute of the library name component is not set. This exception is used by the LName and LNameComponent classes.

See Also:
Serialized Form

Constructor Summary

NotSet() This is the constructor for the class.
NotSet(java.lang.String msg) This is the constructor for the class.

Methods inherited from class java.lang.Throwable

`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

NotSet

```
public NotSet()
```

This is the constructor for the class.

NotSet

```
public NotSet(java.lang.String msg)
```

This is the constructor for the class.

---

com.beasys.Tobj
Class OverFlow

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--com.beasys.Tobj.OverFlow
```

public class **OverFlow**
extends java.lang.Exception

The library cannot allocate resources for the inserted components. This exception is used by the LName and LNameComponent classes.

See Also:
Serialized Form

Constructor Summary

OverFlow() This is the constructor for the class.	
OverFlow(java.lang.String msg) This is the constructor for the class.	

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

OverFlow

```
public OverFlow()
```

This is the constructor for the class.

OverFlow

```
public OverFlow(java.lang.String msg)
```

This is the constructor for the class.

---

com.beasys.Tobj
Class RegistrarNotAvailable

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.Tobj.RegistrarNotAvailable
```

public final class **RegistrarNotAvailable**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

This exception is raised if the FactoryFinder object cannot locate the CORBAservices Naming Service object. Notify the operations staff immediately if this exception is raised. If no naming services servers are running, restart the application.

See Also:
Serialized Form

Constructor Summary

RegistrarNotAvailable() This is the constructor for the class.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

RegistrarNotAvailable

```
public RegistrarNotAvailable()
```

This is the constructor for the class.

---

com.beasys.Tobj
Class RMfailed

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.Tobj.RMfailed
```

public final class **RMfailed**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

The corresponding call to the tx_open() or tx_close() function failed.

See Also:
Serialized Form

Constructor Summary

RMfailed() This is the constructor for the class.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

RMfailed

```
public RMfailed()
```

This is the constructor for the class.

Package com.beasys.TobjS

Class Summary

DeactivateReasonValue	

Exception Summary	
ActivateObjectFailed	An error occurred during the execution of the object's com.beasys.Tobj_Servant.activate_object method.
ApplicationProblem	An unspecified problem has occurred with the server application.
CannotProceed	The FactoryFinder or CORBAservices Naming Service encountered an internal error during the search, with the error being written to the user log (ULOG).
CreateServantFailed	The TP Framework is unable to create the object's servant class.
DeactivateObjectFailed	An error occurred while the com.beasys.Tobj_Servant.deactivate_object method was invoked.
IllegalOperation	Indicates that the com.beasys.Tobj.TP.deactivateEnable method was invoked by an object with the activation policy set to transaction.
InitializeFailed	If an exception is raised in the com.beasys.Tobj.Server.initialize method, the TP Framework catches the exception.
InvalidDomain	The domain specified in the request is invalid.
InvalidInterface	Indicates that the specified interface_name is null.
InvalidName	Raised if id is not one of the four names allowed (see id parameter).
InvalidObject	Indicates that the factory value is null.
InvalidObjectId	Indicates that the specified stroid is null.
InvalidServant	The servant is invalid.
NilObject	Indicates that the method was invoked outside the scope of an application's CORBA object execution.
NoSuchElement	The specified element cannot be located.
NotFound	An element was not found.
OrbProblem	A problem exists with the Object Request Broker (ORB).
OutOfMemory	The server process ran out of memory.
OverFlow	Indicates that the id string is longer than 128 bytes (currently the maximum allowable length).
RegistrarNotAvailable	Indicates that the FactoryFinder Registrar object cannot locate the CORBAservices Naming Service object.
ReleaseFailed	The execution of the com.beasys.Tobj.Server.release method on the server implementation class failed.
UnknownInterface	The execution of the com.beasys.Tobj.TP.create_object_reference method has failed because an unknown interface ID is specified in the request for an object.

Hierarchy For Package com.beasys.TobjS

Package Hierarchies:
All Packages

Class Hierarchy

- class java.lang.Object
  - class com.beasys.TobjS.**DeactivateReasonValue** (implements org.omg.CORBA.portable.IDLEntity)
  - class java.lang.Throwable (implements java.io.Serializable)
    - class java.lang.Exception
      - class org.omg.CORBA.UserException (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.TobjS.**ActivateObjectFailed** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.TobjS.**ApplicationProblem** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.TobjS.**CannotProceed** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.TobjS.**CreateServantFailed** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.TobjS.**DeactivateObjectFailed** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.TobjS.**IllegalOperation** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.TobjS.**InitializeFailed** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.TobjS.**InvalidDomain** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.TobjS.**InvalidInterface** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.TobjS.**InvalidName** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.TobjS.**InvalidObject** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.TobjS.**InvalidObjectId** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.TobjS.**InvalidServant** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.TobjS.**NilObject** (implements org.omg.CORBA.portable.IDLEntity)
        - class com.beasys.TobjS.**NoSuchElement** (implements

org.omg.CORBA.portable.IDLEntity)

- class com.beasys.TobjS.**NotFound** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**OrbProblem** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**OutOfMemory** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**OverFlow** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**RegistrarNotAvailable** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**ReleaseFailed** (implements org.omg.CORBA.portable.IDLEntity)
- class com.beasys.TobjS.**UnknownInterface** (implements org.omg.CORBA.portable.IDLEntity)

---

com.beasys.TobjS

Class DeactivateReasonValue

```
java.lang.Object
  |
  +--com.beasys.TobjS.DeactivateReasonValue
```

public final class **DeactivateReasonValue**
extends java.lang.Object
implements org.omg.CORBA.portable.IDLEntity

See Also:
Serialized Form

Field Summary

`static DeactivateReasonValue`	**`DR_METHOD_END`** Indicates that the object is being deactivated after the completion of a method.
`static DeactivateReasonValue`	**`DR_SERVER_SHUTDOWN`** Indicates that the object is being deactivated because the server is being shut down in an orderly fashion.
`static DeactivateReasonValue`	**`DR_TRANS_ABORTED`** When the deactivate_object() method is invoked with this reason code, the transaction is marked for rollback only.This reason code is used only for objects that have the transaction activation policy.
`static DeactivateReasonValue`	**`DR_TRANS_COMMITTING`** Indicates that a Current.commit() operation was invoked for the transaction in which the object is involved.

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

DR_METHOD_END

```
public static final DeactivateReasonValue DR_METHOD_END
```

Indicates that the object is being deactivated after the completion of a method. It is used if the object's deactivation policy is:

- method
- transaction (only if there is no transaction in effect)
- process (If TP.deactivateEnable() called.)

DR_SERVER_SHUTDOWN

```
public static final DeactivateReasonValue DR_SERVER_SHUTDOWN
```

Indicates that the object is being deactivated because the server is being shut down in an orderly fashion. It is used if the object's deactivation policy is:

- transaction (only if transaction is active)
- process

Note: when a server is shut down in an orderly fashion, all transactions that the server is involved in are marked for rollback.

DR_TRANS_COMMITTING

```
public static final DeactivateReasonValue DR_TRANS_COMMITTING
```

Indicates that a Current.commit() operation was invoked for the transaction in which the object is involved. This reason code is used only for objects that have the transaction activation policy. It can occur when the transaction is started by either the client or the TP Framework. The deactivate_object() method is invoked just before the transaction manager's two-phase commit algorithm begins; that is, before prepare is sent to the resource managers.

The CORBA object is allowed to vote on the outcome of the transaction when the deactivate_object() method is invoked with the DR_TRANS_COMMITTING reason code. By invoking Current.rollback_only(), the method can force the transaction to be rolled back; otherwise, the two-phase commit algorithm continues. The transaction is not necessarily committed just because the Current.rollback_only() is not invoked in this method. Any other CORBA object or resource manager involved in the transaction could also vote to roll back the transaction.

DR_TRANS_ABORTED

```
public static final DeactivateReasonValue DR_TRANS_ABORTED
```

When the deactivate_object() method is invoked with this reason code, the transaction is marked for rollback only.This reason code is used only for objects that have the transaction activation policy. It can occur when the transaction is started by either the client or automatically by the system.

---

com.beasys.TobjS

Class ActivateObjectFailed

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.ActivateObjectFailed
```

public final class **ActivateObjectFailed**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

An error occurred during the execution of the object's com.beasys.Tobj_Servant.activate_object method. The string "reason" provides descriptive information about the exception.

See Also:
Serialized Form

Field Summary

`java.lang.String`	**`reason`** Provides descriptive information about the exception.

Constructor Summary

`ActivateObjectFailed``()` This is the constructor for the class.
`ActivateObjectFailed``(java.lang.String __reason)` This is the constructor for the class.

Methods inherited from class java.lang.Throwable

`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

reason

```
public java.lang.String reason
```

Provides descriptive information about the exception.

Constructor Detail

ActivateObjectFailed

```
public ActivateObjectFailed()
```

This is the constructor for the class.

ActivateObjectFailed

```
public ActivateObjectFailed(java.lang.String __reason)
```

This is the constructor for the class.

---

com.beasys.TobjS

Class ApplicationProblem

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.ApplicationProblem
```

public final class **ApplicationProblem**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

An unspecified problem has occurred with the server application.

See Also:
Serialized Form

Constructor Summary

`ApplicationProblem``()` This is the constructor for the class.

Methods inherited from class java.lang.Throwable
`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

ApplicationProblem

```
public ApplicationProblem()
```

This is the constructor for the class.

---

com.beasys.TobjS

Class CannotProceed

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.CannotProceed
```

public final class **CannotProceed**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

The FactoryFinder or CORBAservices Naming Service encountered an internal error during the search, with the error being written to the user log (ULOG). Notify the operations staff immediately if this exception is raised. Depending on the severity of the internal error, the server running the FactoryFinder or CORBAservices Naming Service may have terminated. If a FactoryFinder service has terminated, start a new FactoryFinder service. If a CORBAservices Naming Service has terminated and there is another CORBAservices Naming Service running, start a new CORBAservices Naming Service. If no naming services servers are running, restart the application.

See Also:
Serialized Form

Constructor Summary

CannotProceed() This is the constructor for the class.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

CannotProceed

```
public CannotProceed()
```

This is the constructor for the class.

---

com.beasys.TobjS

Class CreateServantFailed

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.CreateServantFailed
```

public final class **CreateServantFailed**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

The TP Framework is unable to create the object's servant class. The string "reason" provides descriptive information about the exception.

See Also:
Serialized Form

Field Summary

`java.lang.String`	**`reason`** Provides descriptive information about the exception.

Constructor Summary

`CreateServantFailed``()` This is the constructor for the class.
`CreateServantFailed``(java.lang.String __reason)` This is the constructor for the class.

Methods inherited from class java.lang.Throwable

`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

reason

```
public java.lang.String reason
```

Provides descriptive information about the exception.

Constructor Detail

CreateServantFailed

```
public CreateServantFailed()
```

This is the constructor for the class.

CreateServantFailed

```
public CreateServantFailed(java.lang.String __reason)
```

This is the constructor for the class.

---

com.beasys.TobjS
Class DeactivateObjectFailed

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.DeactivateObjectFailed
```

public final class **DeactivateObjectFailed**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

An error occurred while the com.beasys.Tobj_Servant.deactivate_object method was invoked. The string "reason" provides descriptive information about the exception.

See Also:
Serialized Form

Field Summary

`java.lang.String`	**`reason`** Provides descriptive information about the exception.

Constructor Summary

`DeactivateObjectFailed``()` This is the constructor for the class.
`DeactivateObjectFailed``(java.lang.String __reason)` This is the constructor for the class.

Methods inherited from class java.lang.Throwable

`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

reason

```
public java.lang.String reason
```

Provides descriptive information about the exception.

Constructor Detail

DeactivateObjectFailed

```
public DeactivateObjectFailed()
```

This is the constructor for the class.

DeactivateObjectFailed

```
public DeactivateObjectFailed(java.lang.String __reason)
```

This is the constructor for the class.

---

com.beasys.TobjS
Class IllegalOperation

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.IllegalOperation
```

public final class **IllegalOperation**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

Indicates that the com.beasys.Tobj.TP.deactivateEnable method was invoked by an object with the activation policy set to transaction.

See Also:
Serialized Form

Constructor Summary

`IllegalOperation``()` This is the constructor for the class.	

Methods inherited from class java.lang.Throwable
`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

IllegalOperation

`public` **`IllegalOperation`**`()`

This is the constructor for the class.

---

com.beasys.TobjS
Class InitializeFailed

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.InitializeFailed
```

public final class **InitializeFailed**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

If an exception is raised in the com.beasys.Tobj.Server.initialize method, the TP Framework catches the exception. The TP Framework behavior is the same as if the initialize method returned FALSE (that is, an exception is considered to be a failure). In addition, an error message is written to the user log (ULOG) file. For more information, see the System Messages manual. The string "reason" provides descriptive information about the exception.

See Also:
Serialized Form

Field Summary

java.lang.String	**reason** Provides more descriptive information about the exception.

Constructor Summary

InitializeFailed() This is the constructor for the class.
InitializeFailed(java.lang.String __reason) This is the constructor for the class.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

reason

```
public java.lang.String reason
```

Provides more descriptive information about the exception.

Constructor Detail

InitializeFailed

```
public InitializeFailed()
```

This is the constructor for the class.

InitializeFailed

```
public InitializeFailed(java.lang.String __reason)
```

This is the constructor for the class.

---

com.beasys.TobjS

Class InvalidDomain

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.InvalidDomain
```

public final class **InvalidDomain**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

The domain specified in the request is invalid.

See Also:
Serialized Form

Constructor Summary

`InvalidDomain``()` This is the constructor for the class.

Methods inherited from class java.lang.Throwable
`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

InvalidDomain

```
public InvalidDomain()
```

This is the constructor for the class.

---

com.beasys.TobjS
Class InvalidInterface

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.InvalidInterface
```

public final class **InvalidInterface**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

Indicates that the specified interface_name is null.

See Also:
Serialized Form

Constructor Summary

`InvalidInterface``()` This is the constructor for the class.

Methods inherited from class java.lang.Throwable
`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

InvalidInterface

```
public InvalidInterface()
```

This is the constructor for the class.

---

com.beasys.TobjS
Class InvalidName

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.InvalidName
```

public final class **InvalidName**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

Raised if id is not one of the four names allowed (see id parameter). On the server, also raised when SecurityCurrent is passed.

See Also:
Serialized Form

Constructor Summary

`InvalidName()` This is the constructor for the class.

Methods inherited from class java.lang.Throwable
`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

InvalidName

```
public InvalidName()
```

This is the constructor for the class.

---

com.beasys.TobjS

Class InvalidObject

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.InvalidObject
```

public final class **InvalidObject**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

Indicates that the factory value is null.

See Also:
Serialized Form

Constructor Summary

`InvalidObject``()` This is the constructor for the class.

Methods inherited from class java.lang.Throwable
`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

InvalidObject

```
public InvalidObject()
```

This is the constructor for the class.

---

com.beasys.TobjS

Class InvalidObjectId

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.InvalidObjectId
```

public final class **InvalidObjectId**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

Indicates that the specified stroid is null.

See Also:
Serialized Form

Constructor Summary

`InvalidObjectId``()` This is the constructor for the class.

Methods inherited from class java.lang.Throwable
`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

InvalidObjectId

`public` **`InvalidObjectId`**`()`

This is the constructor for the class.

---

com.beasys.TobjS
Class InvalidServant

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.InvalidServant
```

public final class **InvalidServant**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

The servant is invalid.

See Also:
Serialized Form

Constructor Summary

`InvalidServant()` This is the constructor for the class.

Methods inherited from class java.lang.Throwable
`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

InvalidServant

```
public InvalidServant()
```

This is the constructor for the class.

---

com.beasys.TobjS

Class NilObject

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.NilObject
```

public final class **NilObject**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

Indicates that the method was invoked outside the scope of an application's CORBA object execution. The reason string contains OutOfScope. The string "reason" provides descriptive information about the exception.

See Also:
Serialized Form

Field Summary

java.lang.String	**reason** Provides descriptive information about the exception.

Constructor Summary

NilObject() This is the constructor for the class.
NilObject(java.lang.String __reason) This is the constructor for the class.

Methods inherited from class java.lang.Throwable

fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

reason

```
public java.lang.String reason
```

Provides descriptive information about the exception.

Constructor Detail

NilObject

```
public NilObject()
```

This is the constructor for the class.

NilObject

```
public NilObject(java.lang.String __reason)
```

This is the constructor for the class.

---

com.beasys.TobjS
Class NoSuchElement

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.NoSuchElement
```

public final class **NoSuchElement**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

The specified element cannot be located.

See Also:
Serialized Form

Constructor Summary

`NoSuchElement()` This is the constructor for the class.

Methods inherited from class java.lang.Throwable
`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

NoSuchElement

```
public NoSuchElement()
```

This is the constructor for the class.

---

com.beasys.TobjS

Class NotFound

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.NotFound
```

public final class **NotFound**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

An element was not found.

See Also:
Serialized Form

Constructor Summary

NotFound() This is the constructor for the class.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

NotFound

```
public NotFound()
```

This is the constructor for the class.

---

com.beasys.TobjS

Class OrbProblem

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.OrbProblem
```

public final class **OrbProblem**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

A problem exists with the Object Request Broker (ORB).

See Also:
Serialized Form

Constructor Summary

OrbProblem() This is the constructor for the class.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

OrbProblem

```
public OrbProblem()
```

This is the constructor for the class.

---

com.beasys.TobjS
Class OutOfMemory

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.OutOfMemory
```

public final class **OutOfMemory**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

The server process ran out of memory.

See Also:
Serialized Form

Constructor Summary

`OutOfMemory()` This is the constructor for the class.

Methods inherited from class java.lang.Throwable
`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

OutOfMemory

```
public OutOfMemory()
```

This is the constructor for the class.

---

com.beasys.TobjS
Class OverFlow

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.OverFlow
```

public final class **OverFlow**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

Indicates that the id string is longer than 128 bytes (currently the maximum allowable length).

See Also:
Serialized Form

Constructor Summary

OverFlow() This is the constructor for the class.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

OverFlow

```
public OverFlow()
```

This is the constructor for the class.

---

com.beasys.TobjS

Class RegistrarNotAvailable

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.RegistrarNotAvailable
```

public final class **RegistrarNotAvailable**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

Indicates that the FactoryFinder Registrar object cannot locate the CORBAservices Naming Service object. Notify the operations staff immediately if this exception is raised. If no naming services servers are running, restart the application.

See Also:
Serialized Form

Constructor Summary

RegistrarNotAvailable() This is the constructor for the class.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

RegistrarNotAvailable

```
public RegistrarNotAvailable()
```

This is the constructor for the class.

---

com.beasys.TobjS
Class ReleaseFailed

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.ReleaseFailed
```

public final class **ReleaseFailed**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

The execution of the com.beasys.Tobj.Server.release method on the server implementation class failed. The string "reason" provides descriptive information about the exception.

See Also:
Serialized Form

Field Summary

`java.lang.String`	**`reason`** Provides descriptive information about the exception.

Constructor Summary

`ReleaseFailed``()` This is the constructor for the class.
`ReleaseFailed``(java.lang.String __reason)` This is the constructor for the class.

Methods inherited from class java.lang.Throwable

`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

reason

```
public java.lang.String reason
```

Provides descriptive information about the exception.

Constructor Detail

ReleaseFailed

```
public ReleaseFailed()
```

This is the constructor for the class.

ReleaseFailed

```
public ReleaseFailed(java.lang.String __reason)
```

This is the constructor for the class.

---

com.beasys.TobjS

Class UnknownInterface

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--com.beasys.TobjS.UnknownInterface
```

public final class **UnknownInterface**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

The execution of the com.beasys.Tobj.TP.create_object_reference method has failed because an unknown interface ID is specified in the request for an object.

See Also:
Serialized Form

Constructor Summary

`UnknownInterface``()` This is the constructor for the class.

Methods inherited from class java.lang.Throwable
`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

UnknownInterface

`public` **`UnknownInterface`**`()`

This is the constructor for the class.

Package javax.transaction

Interface Summary	
UserTransaction	The UserTransaction interface defines the methods that allow an application to explicitly manage transaction boundaries.

Exception Summary	
HeuristicCommitException	This exception is thrown by the rollback operation on a resource to report that a heuristic decision was made and that all relevant updates have been committed.
HeuristicException	This exception indicates that one or more participants in a transaction has made a unilateral decision to commit or roll back updates without first obtaining the outcome determined by the transaction service.
HeuristicMixedException	This exception is thrown to report that a heuristic decision was made and that some relevant updates have been committed and others have been rolled back.
HeuristicRollbackException	This exception is thrown by the commit operation to report that a heuristic decision was made and that all relevant updates have been rolled back.
InvalidTransactionException	This exception indicates that the request carried an invalid transaction context.
TransactionRequiredException	This exception indicates that a request carried a null transaction context, but the target object requires an activate transaction.
TransactionRolledbackException	This exception indicates that the transaction associated with processing of the request has been rolled back, or marked to roll back.

Hierarchy For Package javax.transaction

Package Hierarchies:
All Packages

Class Hierarchy

- class java.lang.Object
  - class java.lang.Throwable (implements java.io.Serializable)
    - class java.lang.Exception
      - class java.io.IOException
        - class java.rmi.RemoteException
          - class javax.transaction.**HeuristicCommitException**
          - class javax.transaction.**HeuristicException**
          - class javax.transaction.**HeuristicMixedException**
          - class javax.transaction.**HeuristicRollbackException**
          - class javax.transaction.**InvalidTransactionException**
          - class javax.transaction.**TransactionRequiredException**
          - class javax.transaction.**TransactionRolledbackException**

Interface Hierarchy

- interface javax.transaction.**UserTransaction**

javax.transaction
Interface UserTransaction

public abstract interface **UserTransaction**

The UserTransaction interface defines the methods that allow an application to explicitly manage transaction boundaries.

Field Summary

static int	**STATUS_ACTIVE** A transaction is associated with the target object and it is in the active state.
static int	**STATUS_COMMITTED** A transaction is associated with the target object and it has been committed.
static int	**STATUS_COMMITTING** A transaction is associated with the target object and it is in the process of committing.
static int	**STATUS_MARKED_ROLLBACK** A transaction is associated with the target object and it has been marked for rollback, perhaps as a result of a setRollbackOnly operation.
static int	**STATUS_NO_TRANSACTION** No transaction is currently associated with the target object.
static int	**STATUS_PREPARED** A transaction is associated with the target object and it has been prepared, i.e.
static int	**STATUS_PREPARING** A transaction is associated with the target object and it is in the process of preparing.
static int	**STATUS_ROLLEDBACK** A transaction is associated with the target object and the outcome has been determined as rollback.
static int	**STATUS_ROLLING_BACK** A transaction is associated with the target object and it is in the process of rolling back.
static int	**STATUS_UNKNOWN** A transaction is associated with the target object but its current status cannot be determined.

Method Summary

void	**begin**() Create a new transaction and associate it with the current thread.
void	**commit**() Complete the transaction associated with the current thread.
int	**getStatus**() Obtain the status of the transaction associated with the current thread.
void	**rollback**() Roll back the transaction associated with the current thread.
void	**setRollbackOnly**() Modify the transaction associated with the current thread such that the only possible outcome of the transaction is to roll back the transaction.
void	**setTransactionTimeout**(int seconds) Modify the value of the timeout value that is associated with the transactions started by the current thread with the begin method.

Field Detail

STATUS_ACTIVE

```
public static final int STATUS_ACTIVE
```

A transaction is associated with the target object and it is in the active state. An implementation returns this status after a transaction has been started and prior to a Coordinator issuing any prepares unless the transaction has been marked for rollback.

STATUS_MARKED_ROLLBACK

```
public static final int STATUS_MARKED_ROLLBACK
```

A transaction is associated with the target object and it has been marked for rollback, perhaps as a result of a setRollbackOnly operation.

STATUS_PREPARED

```
public static final int STATUS_PREPARED
```

A transaction is associated with the target object and it has been prepared, i.e. all subordinates have responded Vote.Commit. The target object may be waiting for a superior's instruction as how to proceed.

STATUS_COMMITTED

```
public static final int STATUS_COMMITTED
```

A transaction is associated with the target object and it has been committed. It is likely that heuristics exists, otherwise the transaction would have been destroyed and NoTransaction returned.

STATUS_ROLLEDBACK

```
public static final int STATUS_ROLLEDBACK
```

A transaction is associated with the target object and the outcome has been determined as rollback. It is likely that heuristics exist, otherwise the transaction would have been destroyed and NoTransaction returned.

STATUS_UNKNOWN

```
public static final int STATUS_UNKNOWN
```

A transaction is associated with the target object but its current status cannot be determined. This is a transient condition and a subsequent invocation will ultimately return a different status.

STATUS_NO_TRANSACTION

```
public static final int STATUS_NO_TRANSACTION
```

No transaction is currently associated with the target object. This will occur after a transaction has completed.

STATUS_PREPARING

```
public static final int STATUS_PREPARING
```

A transaction is associated with the target object and it is in the process of preparing. An implementation returns this status if it has started preparing, but has not yet completed the process, probably because it is waiting for responses to prepare from one or more Resources.

STATUS_COMMITTING

```
public static final int STATUS_COMMITTING
```

A transaction is associated with the target object and it is in the process of committing. An implementation returns this status if it has decided to commit, but has not yet completed the process, probably because it is waiting for responses from one or more Resources.

STATUS_ROLLING_BACK

```
public static final int STATUS_ROLLING_BACK
```

A transaction is associated with the target object and it is in the process of rolling back. An implementation returns this status if it has decided to rollback, but has not yet completed the process, probably because it is waiting for responses from one or more Resources.

Method Detail

begin

```
public void begin()
           throws java.lang.IllegalStateException
```

Create a new transaction and associate it with the current thread.

Throws:

java.lang.IllegalStateException - Thrown if the thread is already associated with a transaction.

---

commit

```
public void commit()
            throws TransactionRolledbackException,
                   HeuristicMixedException,
                   HeuristicRollbackException,
                   java.lang.SecurityException,
                   java.lang.IllegalStateException
```

Complete the transaction associated with the current thread. When this method completes, the thread becomes associated with no transaction.

Throws:

TransactionRolledbackException - Thrown to indicate that the transaction has been rolled back rather than committed.

HeuristicMixedException - Thrown to indicate that a heuristic decision was made and that some relevant updates have been committed while others have been rolled back.

HeuristicRollbackException - Thrown to indicate that a heuristic decision was made and that some relevant updates have been rolled back.

java.lang.SecurityException - Thrown to indicate that the thread is not allowed to commit the transaction.

java.lang.IllegalStateException - Thrown if the current thread is not associated with a transaction.

---

rollback

```
public void rollback()
              throws java.lang.IllegalStateException,
                     java.lang.SecurityException
```

Roll back the transaction associated with the current thread. When this method completes, the thread becomes associated with no transaction.

Throws:

java.lang.SecurityException - Thrown to indicate that the thread is not allowed to roll back the transaction.

java.lang.IllegalStateException - Thrown if the current thread is not associated with a transaction.

---

setRollbackOnly

```
public void setRollbackOnly()
                    throws java.lang.IllegalStateException
```

Modify the transaction associated with the current thread such that the only possible outcome of the transaction is to roll back the transaction.

Throws:

java.lang.IllegalStateException - Thrown if the current thread is not associated with a transaction.

---

getStatus

```
public int getStatus()
```

Obtain the status of the transaction associated with the current thread.

Returns:

The transaction status. If no transaction is associated with the current thread, this method returns the Status.NoTransaction value.

---

setTransactionTimeout

```
public void setTransactionTimeout(int seconds)
```

Modify the value of the timeout value that is associated with the transactions started by the current thread with the begin method.

If an application has not called this method, the transaction service uses some default value for the transaction timeout.

Parameters:

`seconds` - The value of the timeout in seconds. If the value is zero, the transaction service restores the default value.

---

javax.transaction
Class HeuristicCommitException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--java.rmi.RemoteException
                          |
                          +--javax.transaction.HeuristicCommitException
```

public class **HeuristicCommitException**
extends java.rmi.RemoteException

This exception is thrown by the rollback operation on a resource to report that a heuristic decision was made and that all relevant updates have been committed.

See Also:
Serialized Form

Fields inherited from class java.rmi.RemoteException
`detail`

Constructor Summary

`HeuristicCommitException``()`	
`HeuristicCommitException``(java.lang.String msg)`	

Methods inherited from class java.rmi.RemoteException
`getMessage, printStackTrace, printStackTrace, printStackTrace`

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

HeuristicCommitException

public **HeuristicCommitException**()

HeuristicCommitException

public **HeuristicCommitException**(java.lang.String msg)

---

javax.transaction
Class HeuristicException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--java.rmi.RemoteException
                          |
                          +--javax.transaction.HeuristicException
```

public class **HeuristicException**
extends java.rmi.RemoteException

This exception indicates that one or more participants in a transaction has made a unilateral decision to commit or roll back updates without first obtaining the outcome determined by the transaction service.

Heuristic decisions are normally made only in unusual circumstances, such as communication failures, that prevent normal processing. When a participant makes a heuristic decision, there is a risk that the decision will differ from the consensus outcome, potentially resulting in loss of data integrity.

The subclasses of this exception provide more specific reporting of the incorrect heuristic decision or the possibility of incorrect heuristic decision.

See Also:
Serialized Form

Fields inherited from class java.rmi.RemoteException
`detail`

Constructor Summary

`HeuristicException``()`	
`HeuristicException``(java.lang.String msg)`	

Methods inherited from class java.rmi.RemoteException
getMessage, printStackTrace, printStackTrace, printStackTrace

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

HeuristicException

public **HeuristicException**()

HeuristicException

public **HeuristicException**(java.lang.String msg)

---

javax.transaction
Class HeuristicMixedException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--java.rmi.RemoteException
                          |
                          +--javax.transaction.HeuristicMixedException
```

public class **HeuristicMixedException**
extends java.rmi.RemoteException

This exception is thrown to report that a heuristic decision was made and that some relevant updates have been committed and others have been rolled back.

See Also:
Serialized Form

Fields inherited from class java.rmi.RemoteException
`detail`

Constructor Summary

`HeuristicMixedException``()`	
`HeuristicMixedException``(java.lang.String msg)`	

Methods inherited from class java.rmi.RemoteException
`getMessage, printStackTrace, printStackTrace, printStackTrace`

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

HeuristicMixedException

public **HeuristicMixedException**()

HeuristicMixedException

public **HeuristicMixedException**(java.lang.String msg)

---

javax.transaction
Class HeuristicRollbackException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--java.rmi.RemoteException
                          |
                          +--javax.transaction.HeuristicRollbackException
```

public class **HeuristicRollbackException**
extends java.rmi.RemoteException

This exception is thrown by the commit operation to report that a heuristic decision was made and that all relevant updates have been rolled back.

See Also:
Serialized Form

Fields inherited from class java.rmi.RemoteException
`detail`

Constructor Summary

`HeuristicRollbackException``()`	
`HeuristicRollbackException``(java.lang.String msg)`	

Methods inherited from class java.rmi.RemoteException
`getMessage, printStackTrace, printStackTrace, printStackTrace`

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

HeuristicRollbackException

```
public HeuristicRollbackException()
```

HeuristicRollbackException

```
public HeuristicRollbackException(java.lang.String msg)
```

---

javax.transaction
Class InvalidTransactionException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--java.rmi.RemoteException
                          |
                          +--javax.transaction.InvalidTransactionException
```

public class **InvalidTransactionException**
extends java.rmi.RemoteException

This exception indicates that the request carried an invalid transaction context. For example, this exception could be raised if an error occurred when trying to register a resource.

See Also:
Serialized Form

Fields inherited from class java.rmi.RemoteException
`detail`

Constructor Summary

`InvalidTransactionException``()`	
`InvalidTransactionException``(java.lang.String msg)`	

Methods inherited from class java.rmi.RemoteException
`getMessage, printStackTrace, printStackTrace, printStackTrace`

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

InvalidTransactionException

public **InvalidTransactionException**()

InvalidTransactionException

public **InvalidTransactionException**(java.lang.String msg)

---

javax.transaction
Class TransactionRequiredException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--java.rmi.RemoteException
                          |
                          +--javax.transaction.TransactionRequiredException
```

public class **TransactionRequiredException**
extends java.rmi.RemoteException

This exception indicates that a request carried a null transaction context, but the target object requires an activate transaction.

See Also:
Serialized Form

Fields inherited from class java.rmi.RemoteException
`detail`

Constructor Summary

`TransactionRequiredException``()`	
`TransactionRequiredException``(java.lang.String msg)`	

Methods inherited from class java.rmi.RemoteException
`getMessage, printStackTrace, printStackTrace, printStackTrace`

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

TransactionRequiredException

public **TransactionRequiredException**()

TransactionRequiredException

public **TransactionRequiredException**(java.lang.String msg)

---

javax.transaction
Class TransactionRolledbackException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.io.IOException
                    |
                    +--java.rmi.RemoteException
                          |
                          +--javax.transaction.TransactionRolledbackException
```

public class **TransactionRolledbackException**
extends java.rmi.RemoteException

This exception indicates that the transaction associated with processing of the request has been rolled back, or marked to roll back. Thus the requested operation either could not be performed or was not performed because further computation on behalf of the transaction would be fruitless

See Also:
Serialized Form

Fields inherited from class java.rmi.RemoteException
`detail`

Constructor Summary

`TransactionRolledbackException``()`	
`TransactionRolledbackException``(java.lang.String msg)`	

Methods inherited from class java.rmi.RemoteException
`getMessage, printStackTrace, printStackTrace, printStackTrace`

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

TransactionRolledbackException

public **TransactionRolledbackException**()

TransactionRolledbackException

public **TransactionRolledbackException**(java.lang.String msg)

---

org.omg.CosLifeCycle
Interface FactoryFinder

All Known Subinterfaces:
FactoryFinder

public abstract interface **FactoryFinder**
extends org.omg.CORBA.Object, org.omg.CORBA.portable.IDLEntity

Fields inherited from class java.io.Serializable
`serialVersionUID`

Method Summary	
`org.omg.CORBA.Object[]`	**`find_factories`**`(org.omg.CosNaming.NameComponent[] factory_key)` Obtains a list of application factories.

Methods inherited from interface org.omg.CORBA.Object
`_create_request, _create_request, _duplicate, _get_domain_managers, _get_interface_def, _get_policy, _hash, _is_a, _is_equivalent, _non_existent, _release, _request, _set_policy_override`

Method Detail

find_factories

```
public org.omg.CORBA.Object[] find_factories(org.omg.CosNaming.NameComponent[] factory_key)
                                     throws org.omg.CosLifeCycle.NoFactory
```

Obtains a list of application factories. The find_factories method instructs the FactoryFinder to return a list of server application factory object references whose keys match the method's input key. The M3 system assumes that an equality match is to be performed. This means that for the two sequences of pairs (those corresponding to the input key and those in the application factories keys), each are of equal length; and for every pair in one sequence, there is an identical pair in the other.

Parameters:

`factory_key` - This parameter contains a sequence of NameComponents (value pairs) that uniquely identifies a factory object reference.

Returns:

Returns a sequence of object references for application factories.

Throws:

org.omg.CosLifeCycle.NoFactory - This exception is raised if the FactoryFinder cannot find an application factory object reference that corresponds to the input factory_key.

Package org.omg.CosTransactions

Interface Summary	
Control	The Control interface allows a program to explicitly manage or propagate a transaction context.
Current	The Current interface defines methods that allow a client of the Transaction Service to explicitly manage the association between threads and transactions.

Exception Summary	
HeuristicHazard	A request raises this exception to report that a heuristic decision was made, the disposition of all relevant updates is not known, and for those updates whose disposition is known, either all have been committed or all have been rolled back.
HeuristicMixed	A request raises this exception to report that a heuristic decision was made and that some relevant updates have been committed and others have been rolled back.
InvalidControl	This exception is raised for the Current interface resume method if the parameter is not valid in the current execution environment.
NoTransaction	This exception is raised for the Current interface commit, rollback, and rollback_only methods if there is no transaction associated with the client thread.
SubtransactionsUnavailable	This exception is raised for the Current interface begin method if the client already has an associated transaction and the Transaction Service implementation does not support nested transactions.
Unavailable	This exception is raised for the Control interface get_terminator and get_coordinator methods if the Control cannot provide the requested object.

Hierarchy For Package org.omg.CosTransactions

Package Hierarchies:
All Packages

Class Hierarchy

- class java.lang.Object
  - class java.lang.Throwable (implements java.io.Serializable)
    - class java.lang.Exception
      - class org.omg.CORBA.UserException (implements org.omg.CORBA.portable.IDLEntity)
        - class org.omg.CosTransactions.**HeuristicHazard** (implements org.omg.CORBA.portable.IDLEntity)
        - class org.omg.CosTransactions.**HeuristicMixed** (implements org.omg.CORBA.portable.IDLEntity)
        - class org.omg.CosTransactions.**InvalidControl** (implements org.omg.CORBA.portable.IDLEntity)
        - class org.omg.CosTransactions.**NoTransaction** (implements org.omg.CORBA.portable.IDLEntity)
        - class org.omg.CosTransactions.**SubtransactionsUnavailable** (implements org.omg.CORBA.portable.IDLEntity)
        - class org.omg.CosTransactions.**Unavailable** (implements org.omg.CORBA.portable.IDLEntity)

Interface Hierarchy

- interface org.omg.CORBA.Object
  - interface org.omg.CosTransactions.**Control**(also extends org.omg.CORBA.portable.IDLEntity)
  - interface org.omg.CosTransactions.**Current**(also extends org.omg.CORBA.portable.IDLEntity)
- interface java.io.Serializable
  - interface org.omg.CORBA.portable.IDLEntity
    - interface org.omg.CosTransactions.**Control**(also extends org.omg.CORBA.Object)
    - interface org.omg.CosTransactions.**Current**(also extends org.omg.CORBA.Object)

org.omg.CosTransactions
Interface Control

public abstract interface **Control**
extends org.omg.CORBA.Object, org.omg.CORBA.portable.IDLEntity

The Control interface allows a program to explicitly manage or propagate a transaction context. An object that supports the Control interface is implicitly associated with one specific transaction.

Fields inherited from class java.io.Serializable
`serialVersionUID`

Method Summary

`Coordinator`	**`get_coordinator`**`()`
`Terminator`	**`get_terminator`**`()` Returns a Terminator object, which supports methods to end the transaction.

Methods inherited from interface org.omg.CORBA.Object
`_create_request, _create_request, _duplicate, _get_domain_managers, _get_interface_def, _get_policy, _hash, _is_a, _is_equivalent, _non_existent, _release, _request, _set_policy_override`

Method Detail

get_terminator

```
public Terminator get_terminator()
                          throws Unavailable
```

Returns a Terminator object, which supports methods to end the transaction. The object can be used to roll back or commit the transaction associated with the Control interface. This implementation of the Transaction Service always raises the Unavailable exception.

get_coordinator

```
public Coordinator get_coordinator()
                            throws Unavailable
```

org.omg.CosTransactions
Interface Current

All Known Subinterfaces:
TransactionCurrent

public abstract interface **Current**
extends org.omg.CORBA.Object, org.omg.CORBA.portable.IDLEntity

The Current interface defines methods that allow a client of the Transaction Service to explicitly manage the association between threads and transactions. This interface also defines methods that simplify the use of the Transaction Service for most applications. These methods can be used to demarcate transactions, to suspend/resume transactions, and to obtain information about the current transaction.

Fields inherited from class java.io.Serializable
`serialVersionUID`

Method Summary

void	**begin**() Creates a new transaction.
void	**commit**(boolean report_heuristics) Completes the current transaction.
Control	**get_control**() If the client is associated with a transaction, a Control object is returned that represents the transaction context currently associated with the client thread.
Status	**get_status**() If there is a transaction associated with the client thread, this method returns the status of the transaction associated with the client thread.
java.lang.String	**get_transaction_name**() If there is a transaction associated with the client thread, this method returns a printable string describing the transaction (specifically the XID as specified by the Open Group).
void	**resume**(Control which) If the parameter is a valid object reference in the current execution environment, the client thread becomes associated with that transaction (in place of any previous transaction).
void	**rollback_only**() If there is a transaction associated with the client thread, it is modified so that the only possible outcome is to roll back the transaction.
void	**rollback**() Rolls back the current transaction.
void	**set_timeout**(int seconds) This method modifies a state variable associated with the target object that affects the time-out period associated with transactions created by subsequent invocations of the begin method.
Control	**suspend**() If in a proper context with a transaction not marked ROLLEDBACK, an object is returned that represents the transaction context currently associated with the client thread.

Methods inherited from interface org.omg.CORBA.Object
_create_request, _create_request, _duplicate, _get_domain_managers, _get_interface_def, _get_policy, _hash, _is_a, _is_equivalent, _non_existent, _release, _request, _set_policy_override

Method Detail

begin

```
public void begin()
           throws SubtransactionsUnavailable
```

Creates a new transaction. The transaction context of the client thread is modified so that the thread is associated with the new transaction.

If the client thread is currently associated with a transaction, the SubtransactionsUnavailable exception is raised.

If the client thread cannot be placed in transaction mode due to an error while starting the transaction, the org.omg.CORBA.INVALID_TRANSACTION standard system exception is raised.

If the call was made in an improper context, the org.omg.CORBA.BAD_INV_ORDER standard system exception is raised.

---

commit

```
public void commit(boolean report_heuristics)
            throws NoTransaction,
                   HeuristicMixed,
                   HeuristicHazard
```

Completes the current transaction. If the operation completes normally, the thread's transaction context is set to null. If there is no transaction associated with the client thread, the org.omg.CosTransactions.NoTransaction exception is raised.

If the call was made in an improper context, the standard system exception org.omg.CORBA.BAD_INV_ORDER is raised.

If the system decides to roll back the transaction, the standard system exception org.omg.CORBA.TRANSACTION_ROLLEDBACK is raised and the thread's transaction context is set to null.

An org.omg.CosTransactions.HeuristicMixed exception is raised to report that a heuristic decision was made and that some relevant updates have been committed and others have been rolled back. A HeuristicHazard exception is raised to report that a heuristic decision was made, and the disposition of all relevant updates is not known. For those updates whose disposition is known, either all have been committed or all have been rolled back. The HeuristicMixed exception takes priority over the HeuristicHazard exception. If a heuristic exception is raised or the operation completes normally, the thread's transaction context is set to null.

---

rollback

```
public void rollback()
              throws NoTransaction
```

Rolls back the current transaction. A return from the rollback method on the Current object is asynchronous. A consequence of this is that the objects that were infected by the rolled back transaction get their states cleared by the M3 TP Framework *a little later*. This implies that no other client can infect these objects with a different transaction until the M3 TP Framework clears the states of these objects. This race condition exists for a very short amount of time and is typically not noticeable in a full-fledged application. A simple workaround for this race condition is to try the appropriate operation after a short (typically a 1-second) delay.

If there is no transaction associated with the client thread, the org.omg.CosTransactions.NoTransaction exception is raised.

If the call was made in an improper context, the standard system exception org.omg.CORBA.BAD_INV_ORDER is raised.

If the operation completes normally, the thread's transaction context is set to null.

---

rollback_only

```
public void rollback_only()
                  throws NoTransaction
```

If there is a transaction associated with the client thread, it is modified so that the only possible outcome is to roll back the transaction. Otherwise, the NoTransaction exception is raised.

---

get_status

```
public Status get_status()
```

If there is a transaction associated with the client thread, this method returns the status of the transaction associated with the client thread. Otherwise, the StatusNoTransaction value is returned.

---

get_transaction_name

```
public java.lang.String get_transaction_name()
```

If there is a transaction associated with the client thread, this method returns a printable string describing the transaction (specifically the XID as specified by the Open Group). Otherwise, an empty string is returned. The returned string is intended to support debugging.

---

set_timeout

```
public void set_timeout(int seconds)
```

This method modifies a state variable associated with the target object that affects the time-out period associated with transactions created by subsequent invocations of the begin method. If the parameter has a nonzero value n, transactions created by subsequent invocation of begin will be subject to being rolled back if they do not complete before n seconds after their creation. If the parameter is zero, no application-specified time-out is established.

If the time-out period is not defined, the M3 system uses a default value, which is specified in the UBBCONFIG file. For more information, see the description of the TIMEOUT parameter in the Administration Guide.

Note: The initial transaction timeout value is 300 seconds. If a transaction is started via AUTOTRAN instead of via the begin method, the timeout value is determined by the TRANTIME value in the M3 configuration file. For more information, refer to Chapter 7 of the Administration Guide.

---

get_control

```
public Control get_control()
```

If the client is associated with a transaction, a Control object is returned that represents the transaction context currently associated with the client thread. This object may be given to the resume method to reestablish this context. If the client is not associated with a transaction, a null object reference is returned.

---

suspend

```
public Control suspend()
```

If in a proper context with a transaction not marked ROLLEDBACK, an object is returned that represents the transaction context currently associated with the client thread. The same client can subsequently give this object to the resume method to reestablish this context. In addition, the client thread becomes associated with no transaction. If the client thread is not associated with a transaction, a null object reference is returned.

If the associated transaction is in a state such that the only possible outcome of the transaction is to be rolled back, the org.omg.CORBA.TRANSACTION_ROLLEDBACK standard system exception is raised and the client thread becomes associated with no transaction.

If the call was made in an improper context, the standard system exception org.omg.CORBA.BAD_INV_ORDER is raised. The caller’s state with respect to the transaction is not changed.

Note: As defined in The Common Object Request Broker: Architecture and Specification, Revision 2.2, February 1998, the org.omg.CORBA.TRANSACTION_ROLLEDBACK standard system exception indicates that the transaction associated with the request has already been rolled back or has been marked to roll back. Thus, the requested method either could not be performed or was not performed because further computation on behalf of the transaction would be fruitless.

resume

```
public void resume(Control which)
            throws InvalidControl
```

If the parameter is a valid object reference in the current execution environment, the client thread becomes associated with that transaction (in place of any previous transaction). If the client thread is already associated with a transaction that is in a state such that the only possible outcome of the transaction is to be rolled back, the org.omg.CORBA.TRANSACTION_ROLLEDBACK standard system exception is raised and the client thread becomes associated with no transaction. If the parameter is not valid, the org.omg.CosTransactions.InvalidControl exception is raised.

If the call was made in an improper context, the standard system exception org.omg.CORBA.BAD_INV_ORDER is raised.

If the system is unable to resume the global transaction because the caller is currently participating in work outside any global transaction with one or more resource managers, the org.omg.CORBA.INVALID_TRANSACTION standard system exception is raised.

Note: See suspend for a definition of the org.omg.CORBA.TRANSACTION_ROLLEDBACK standard system exception.

org.omg.CosTransactions
Class HeuristicHazard

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--org.omg.CosTransactions.HeuristicHazard
```

public final class **HeuristicHazard**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

A request raises this exception to report that a heuristic decision was made, the disposition of all relevant updates is not known, and for those updates whose disposition is known, either all have been committed or all have been rolled back. Therefore, the org.omg.CosTransactions.HeuristicMixed exception takes priority over the HeuristicHazard exception.

See Also:
Serialized Form

Constructor Summary

HeuristicHazard() This is the constructor for the class.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

HeuristicHazard

```
public HeuristicHazard()
```

This is the constructor for the class.

---

org.omg.CosTransactions
Class HeuristicMixed

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--org.omg.CosTransactions.HeuristicMixed
```

public final class **HeuristicMixed**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

A request raises this exception to report that a heuristic decision was made and that some relevant updates have been committed and others have been rolled back.

See Also:
Serialized Form

Constructor Summary

HeuristicMixed() This is the constructor for the class.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

HeuristicMixed

```
public HeuristicMixed()
```

This is the constructor for the class.

---

org.omg.CosTransactions
Class InvalidControl

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--org.omg.CosTransactions.InvalidControl
```

public final class **InvalidControl**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

This exception is raised for the Current interface resume method if the parameter is not valid in the current execution environment.

See Also:
Serialized Form

Constructor Summary

`InvalidControl``()` This is the constructor for the class.

Methods inherited from class java.lang.Throwable
`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

InvalidControl

```
public InvalidControl()
```

This is the constructor for the class.

---

org.omg.CosTransactions
Class NoTransaction

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--org.omg.CosTransactions.NoTransaction
```

public final class **NoTransaction**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

This exception is raised for the Current interface commit, rollback, and rollback_only methods if there is no transaction associated with the client thread.

See Also:
Serialized Form

Constructor Summary

NoTransaction() This is the constructor for the class.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

NoTransaction

```
public NoTransaction()
```

This is the constructor for the class.

---

org.omg.CosTransactions
Class SubtransactionsUnavailable

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--org.omg.CosTransactions.SubtransactionsUnavailable
```

public final class **SubtransactionsUnavailable**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

This exception is raised for the Current interface begin method if the client already has an associated transaction and the Transaction Service implementation does not support nested transactions.

See Also:
Serialized Form

Constructor Summary

SubtransactionsUnavailable() This is the constructor for the class.

Methods inherited from class java.lang.Throwable
fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Detail

SubtransactionsUnavailable

```
public SubtransactionsUnavailable()
```

This is the constructor for the class.

---

org.omg.CosTransactions
Class Unavailable

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--org.omg.CORBA.UserException
                    |
                    +--org.omg.CosTransactions.Unavailable
```

public final class **Unavailable**
extends org.omg.CORBA.UserException
implements org.omg.CORBA.portable.IDLEntity

This exception is raised for the Control interface get_terminator and get_coordinator methods if the Control cannot provide the requested object.

See Also:
Serialized Form

Constructor Summary

`Unavailable()` This is the constructor for the class.

Methods inherited from class java.lang.Throwable
`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object
`equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

Unavailable

```
public Unavailable()
```

This is the constructor for the class.

Package org.omg.Security

Class Summary

AuthenticationStatus	

Overview Package Class **Tree Deprecated Index Help**
PREV PACKAGE NEXT PACKAGE **FRAMES NO FRAMES**

Hierarchy For Package org.omg.Security

Package Hierarchies:
All Packages

Class Hierarchy

- class java.lang.Object
  - class org.omg.Security.**AuthenticationStatus** (implements org.omg.CORBA.portable.IDLEntity)

org.omg.Security
Class AuthenticationStatus

```
java.lang.Object
  |
  +--org.omg.Security.AuthenticationStatus
```

public final class **AuthenticationStatus**
extends java.lang.Object
implements org.omg.CORBA.portable.IDLEntity

See Also:
Serialized Form

Field Summary

static int	**_SecAuthContinue** Not used in the M3 system.
static int	**_SecAuthExpired** Not used in the M3 system.
static int	**_SecAuthFailure** The authentication failed, or the client was already authenticated and did not invoke com.beasys.Tobj.PrincipalAuthenticator.logoff() or com.beasys.Tobj_Bootstrap.destroy_current().
static int	**_SecAuthSuccess** The authentication succeeded.
static AuthenticationStatus	**SecAuthContinue** Not used in the M3 system.
static AuthenticationStatus	**SecAuthExpired** Not used in the M3 system.
static AuthenticationStatus	**SecAuthFailure** The authentication failed, or the client was already authenticated and did not invoke com.beasys.Tobj.PrincipalAuthenticator.logoff() or com.beasys.Tobj_Bootstrap.destroy_current().
static AuthenticationStatus	**SecAuthSuccess** The authentication succeeded.

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

_SecAuthSuccess

public static final int **_SecAuthSuccess**

The authentication succeeded.

_SecAuthFailure

public static final int **_SecAuthFailure**

The authentication failed, or the client was already authenticated and did not invoke com.beasys.Tobj.PrincipalAuthenticator.logoff() or com.beasys.Tobj_Bootstrap.destroy_current().

_SecAuthContinue

public static final int **_SecAuthContinue**

Not used in the M3 system.

_SecAuthExpired

public static final int **_SecAuthExpired**

Not used in the M3 system.

SecAuthSuccess

public static final AuthenticationStatus **SecAuthSuccess**

The authentication succeeded.

SecAuthFailure

`public static final AuthenticationStatus` **`SecAuthFailure`**

The authentication failed, or the client was already authenticated and did not invoke com.beasys.Tobj.PrincipalAuthenticator.logoff() or com.beasys.Tobj_Bootstrap.destroy_current().

SecAuthContinue

`public static final AuthenticationStatus` **`SecAuthContinue`**

Not used in the M3 system.

SecAuthExpired

`public static final AuthenticationStatus` **`SecAuthExpired`**

Not used in the M3 system.

Package org.omg.SecurityLevel1

Interface Summary	
Current	

Hierarchy For Package org.omg.SecurityLevel1

Package Hierarchies:
All Packages

Interface Hierarchy

- interface org.omg.CORBA.Object
  - interface org.omg.SecurityLevel1.**Current**(also extends org.omg.CORBA.portable.IDLEntity)
- interface java.io.Serializable
  - interface org.omg.CORBA.portable.IDLEntity
    - interface org.omg.SecurityLevel1.**Current**(also extends org.omg.CORBA.Object)

org.omg.SecurityLevel1
Interface Current

All Known Subinterfaces:
Current

public abstract interface **Current**
extends org.omg.CORBA.Object, org.omg.CORBA.portable.IDLEntity

Fields inherited from class java.io.Serializable
`serialVersionUID`

Method Summary

`SecAttribute[]`	**`get_attributes`**`(AttributeType[] attributes)` Returns attributes for the Current interface.

Methods inherited from interface org.omg.CORBA.Object
`_create_request, _create_request, _duplicate, _get_domain_managers, _get_interface_def, _get_policy, _hash, _is_a, _is_equivalent, _non_existent, _release, _request, _set_policy_override`

Method Detail

get_attributes

`public SecAttribute[] get_attributes(AttributeType[] attributes)`

Returns attributes for the Current interface.

This method gets privilege (and other) attributes from the client's credentials for the Current interface.

Package org.omg.SecurityLevel2

Interface Summary	
Credentials	
Current	
PrincipalAuthenticator	

Overview Package Class **Tree Deprecated Index Help**
PREV PACKAGE NEXT PACKAGE **FRAMES NO FRAMES**

Hierarchy For Package org.omg.SecurityLevel2

Package Hierarchies:
All Packages

Interface Hierarchy

- interface org.omg.CORBA.Object
  - interface org.omg.SecurityLevel2.**Credentials**(also extends org.omg.CORBA.portable.IDLEntity)
  - interface org.omg.SecurityLevel2.**Current**(also extends org.omg.SecurityLevel1.Current, org.omg.CORBA.portable.IDLEntity)
  - interface org.omg.SecurityLevel1.**Current**(also extends org.omg.CORBA.portable.IDLEntity)
    - interface org.omg.SecurityLevel2.**Current**(also extends org.omg.CORBA.portable.IDLEntity, org.omg.CORBA.Object)
  - interface org.omg.SecurityLevel2.**PrincipalAuthenticator**(also extends org.omg.CORBA.portable.IDLEntity)
- interface java.io.Serializable
  - interface org.omg.CORBA.portable.IDLEntity
    - interface org.omg.SecurityLevel2.**Credentials**(also extends org.omg.CORBA.Object)
    - interface org.omg.SecurityLevel2.**Current**(also extends org.omg.SecurityLevel1.Current, org.omg.CORBA.Object)
    - interface org.omg.SecurityLevel1.**Current**(also extends org.omg.CORBA.Object)
      - interface org.omg.SecurityLevel2.**Current**(also extends org.omg.CORBA.portable.IDLEntity, org.omg.CORBA.Object)
    - interface org.omg.SecurityLevel2.**PrincipalAuthenticator**(also extends org.omg.CORBA.Object)

org.omg.SecurityLevel2
Interface Credentials

public abstract interface **Credentials**
extends org.omg.CORBA.Object, org.omg.CORBA.portable.IDLEntity

Fields inherited from class java.io.Serializable
`serialVersionUID`

Method Summary

`SecAttribute[]`	**`get_attributes`**`(AttributeType[] attributes)` Gets the attribute list attached to the credentials.
`boolean`	**`is_valid`**`(org.omg.TimeBase.UtcTHolder expiry_time)` Checks status of credentials.

Methods inherited from interface org.omg.CORBA.Object
`_create_request, _create_request, _duplicate, _get_domain_managers, _get_interface_def, _get_policy, _hash, _is_a, _is_equivalent, _non_existent, _release, _request, _set_policy_override`

Method Detail

get_attributes

`public SecAttribute[] get_attributes(AttributeType[] attributes)`

Gets the attribute list attached to the credentials.

This method returns the attribute list attached to the client’s credentials. In the list of attribute types, you are required to include only the type value(s) for the attributes you want returned in the AttributeList. Attributes are not currently returned based on attribute-family or identities. In most cases, the attribute list returned is the same you would get if you invoked SecurityLevel1.Current.get_attributes(), since there is only one valid set of credentials in the

client at any instance in time. The results could be different if the credentials are not currently in use.

---

is_valid

```
public boolean is_valid(org.omg.TimeBase.UtcTHolder expiry_time)
```

Checks status of credentials.

This method returns TRUE if the credentials used are active at the time; that is, you did not invoke Tobj.PrincipalAuthenticator.logoff() or Tobj_Bootstrap.destroy_current(). If this method is invoked after Tobj.PrincipalAuthenticator.logoff(), FALSE is returned. If this method is invoked after Tobj_Bootstrap.destroy_current(), the org.omg.CORBA.BAD_INV_ORDER exception is raised.

---

org.omg.SecurityLevel2
Interface Current

public abstract interface **Current**
extends org.omg.CORBA.Object, org.omg.CORBA.portable.IDLEntity, Current

Fields inherited from class java.io.Serializable
`serialVersionUID`

Fields inherited from class java.io.Serializable
`serialVersionUID`

Method Summary

`Credentials`	**`get_credentials`**`(CredentialType cred_type)` Gets credentials type.
`PrincipalAuthenticator`	**`principal_authenticator`**`()` Returns the PrincipalAuthenticator object.
`void`	**`set_credentials`**`(CredentialType cred_type, Credentials cred)` Sets credentials type.

Methods inherited from interface org.omg.SecurityLevel1.Current
`get_attributes`

Method Detail

set_credentials

```
public void set_credentials(CredentialType cred_type,
                            Credentials cred)
```

Sets credentials type.

This method can be used only to set SecInvocationCredentials; otherwise, the set_credentials method raises the org.omg.CORBA.BAD_PARAM exception. The credentials must have been obtained from a previous invocation of the SecurityLevel2.Current.get_credentials or SecurityLevel2.PrincipalAuthenticator.authenticate methods.

---

get_credentials

```
public Credentials get_credentials(CredentialType cred_type)
```

Gets credentials type.

This method can be used only to get SecInvocationCredentials; otherwise, the get_credentials method raises the org.omg.CORBA.BAD_PARAM exception. If no credentials are available, the get_credentials method raises the org.omg.CORBA.BAD_INV_ORDER exception.

---

principal_authenticator

```
public PrincipalAuthenticator principal_authenticator()
```

Returns the PrincipalAuthenticator object.

The PrincipalAuthenticator object returned by the principal_authenticator attribute is of actual type Tobj.PrincipalAuthenticator. Therefore, it can be used both as a Tobj.PrincipalAuthenticator and as a SecurityLevel2.PrincipalAuthenticator.

Note: This method raises the org.omg.CORBA.BAD_INV_ORDER exception if it is invoked on an invalid SecurityCurrent object.

---

org.omg.SecurityLevel2
Interface PrincipalAuthenticator

All Known Subinterfaces:
PrincipalAuthenticator

public abstract interface **PrincipalAuthenticator**
extends org.omg.CORBA.Object, org.omg.CORBA.portable.IDLEntity

Fields inherited from class java.io.Serializable
`serialVersionUID`

Method Summary

`AuthenticationStatus`	**`authenticate`**`(int method, java.lang.String security_name, byte[] auth_data, SecAttribute[] privileges, CredentialsHolder creds, OpaqueHolder continuation_data, OpaqueHolder auth_specific_data)` Authenticates the client.
`AuthenticationStatus`	**`continue_authentication`**`(byte[] response_data, CredentialsHolder creds, OpaqueHolder continuation_data, OpaqueHolder auth_specific_data)` Always fails.

Methods inherited from interface org.omg.CORBA.Object
`_create_request, _create_request, _duplicate, _get_domain_managers, _get_interface_def, _get_policy, _hash, _is_a, _is_equivalent, _non_existent, _release, _request, _set_policy_override`

Method Detail

authenticate

```
public AuthenticationStatus authenticate(int method,
                                         java.lang.String security_name,
                                         byte[] auth_data,
                                         SecAttribute[] privileges,
                                         CredentialsHolder creds,
                                         OpaqueHolder continuation_data,
                                         OpaqueHolder auth_specific_data)
```

Authenticates the client.

This method authenticates the client via the IIOP Server Listener/Handler so that it can access an M3 domain.

This method accepts the following input parameters:

- `method` is the integer specified by the com.beasys.Tobj.TuxedoSecurity interface.
- `security_name` is the M3 Java user name.
- `auth_data` is returned by the org.omg.SecurityLevel2.PrincipalAuthenticator.build_auth_data method. If auth_data is invalid, the authenticate method raises the org.omg.CORBA.BAD_PARAM exception.
- `privileges` is of type SecAttribute []. SecAttribute [] is a SecAttribute array. This array can contain an AccessId, which must be the same as the security_name argument (parameter 2), and it can contain a PrimaryGroupId, which is taken to be a TUXEDO client name. In many cases, the privileges are built by an invocation to the build_auth_data method.
- `creds` is the credentials data returned by the org.omg.SecurityLevel2.Current.get_credentials method.
- `continuation_data` and `auth_specific_data` are always empty.

Note: The client can use TUXEDO style authentication and invoke the Tobj.PrincipalAuthenticator.logon method instead of the authenticate method.

---

continue_authentication

```
public AuthenticationStatus continue_authentication(byte[] response_data,
                                                    CredentialsHolder creds,
                                                    OpaqueHolder continuation_data,
                                                    OpaqueHolder auth_specific_data)
```

Always fails.

Because the M3 software does authentication in one step, this method always fails and returns org.omg.Security.AuthenticationStatus.SecAuthFailure.
