OBJECTSTORE

JAVA INTERFACE
RELEASE NOTES

RELEASE 3.0

October 1998

ObjectStore Java Interface Release Notes

ObjectStore Java Interface Release 3.0, October 1998

Object Design, Inc.
Twenty Five Mall Road
Burlington, MA 01803-4194

Contents

Preface

Purpose

ObjectStore Java Interface Release Notes provides information about known problems, requirements for running ObjectStore, compatibility with other ObjectStore Java products, obtaining support, and obtaining third-party persistence-capable classes.

Audience

This book is for experienced Java programmers who are writing applications that use the Java interface to ObjectStore.

Scope

This book supports Release 3.0 of the Java interface to ObjectStore.

This book does not provide information about new features, changes, and bug fixes in this release. For that information, see the **CHANGES.htm** file in the directory in which you installed ObjectStore.

Documentation Conventions

This document uses the following conventions:

Convention	*Meaning*
Bold	Bold typeface indicates user input, code fragments, method signatures, file names, and object, field, and method names.
Sans serif	Sans serif typeface is used for system output and system output.
Italic sans serif	Italic sans serif typeface indicates a variable for which you must supply a value. This most often appears in a syntax line or table.
Italic serif	In text, italic serif typeface indicates the first use of an important term.
[ ]	Brackets enclose optional arguments.
{ *a* | *b* | *c* }	Braces enclose two or more items. You can specify only one of the enclosed items. Vertical bars represent OR separators. For example, you can specify *a* or *b* or *c*.
...	Three consecutive periods indicate that you can repeat the immediately previous item. In examples, they also indicate omissions.

Examples in the documentation

Examples in the documentation assume that **COM.odi.*** is imported. This allows specification of, for example,

db.open(ObjectStore.READONLY)

instead of **db.open(COM.odi.ObjectStore.READONLY)**

Internet Sources of More Information

World Wide Web

Object Design's support organization provides a number of information resources. These are available to you through a web browser such as Internet Explorer or Netscape Navigator. You can obtain information by accessing the Object Design home page with the URL **http://www.objectdesign.com**. Select **Tech Support**. Select **Support Communications** for detailed instructions about different methods of obtaining information from support.

Internet gateway

You can obtain such information as frequently asked questions (FAQs) from Object Design's Internet gateway machine as well as from the web. This machine is called **ftp.objectdesign.com** and its Internet address is 198.3.16.26. You can use **ftp** to retrieve the FAQs from there. Use the login name **objectdesignftp** and the password obtained from **patch-info**. This password also changes monthly, but you can automatically receive the updated password by subscribing to **patch-info**. See the ObjectStore Release 5.1 **README** file for guidelines for using this connection. The FAQs are in the **/support/FAQ** subdirectory. This directory contains a group of subdirectories organized by topic. The **FAQ/FAQ.tar.Z** file is a compressed **tar** version of this hierarchy that you can download.

Automatic email notification

In addition to the previous methods of obtaining Object Design's latest patch updates (available on the **ftp** server as well as the Object Design Support home page), you can now automatically be notified of updates. To subscribe, send email to **majordomo@objectdesign.com** with the keyword **SUBSCRIBE patch-info <***your siteid***>** in the body of your email. This will subscribe you to Object Design's patch information server daemon that automatically provides site access information and notification of other changes to the on-line support services. Your site ID is listed on any shipment from Object Design, or you can contact your Object Design Sales Administrator for the site ID information.

Email discussion list

There is a majordomo discussion list called **osji-discussion**. The purpose of this list is to facilitate discussion about the Java interface to ObjectStore.

Support

Object Design's support organization provides a number of information resources and services. Their home page is at **http://support.objectdesign.com/WWW/Welcome.html**. From the support home page, you can learn about support policies, product discussion groups, and the different ways Object Design can keep you informed about the latest release information — including the Web, **ftp**, and email services.

Training

You can obtain information about training courses from the Object Design web site (**http://www.objectdesign.com**). From the home page, select **Services** and then **Education**.

If you are in North America, for information about Object Design's educational offerings, call 781.674.5000, Monday through Friday from 8:30 AM to 5:30 PM Eastern Time. If you are outside North America, call your Object Design sales representative.

Your Comments

Object Design welcomes your comments about ObjectStore documentation. Send feedback to **support@objectdesign.com**. To expedite your message, begin the subject with **Doc:**. For example:

Subject: Doc: Incorrect message on page 76 of reference manual

You can also fax your comments to 781.674.5440.

Release Notes

For information about new features, changes, and bug fixes in this release, see the **CHANGES.htm** file in the top-level ObjectStore Java interface directory. For information about supported platforms and any last-minute changes, see the **README.htm** file, also in the top-level directory. *Release Notes* provides information about the following topics:

About the release number

There is no Release 2.0 for the Java interface to ObjectStore. OSJI goes from Release 1.3 to Release 3.0 to keep it synchronized with the PSE and PSE Pro for Java releases.

Known Problems and Restrictions

This section describes issues to consider concerning restrictions and known problems:

Threads Are Not Being Automatically Joined to Sessions

Threads that do not belong to a session are not automatically joined to a session when they should be. This includes global sessions. Until this problem is resolved, applications cannot rely on session absorption to make a thread that accesses persistent objects join the appropriate session. As a work around, applications can explicitly join each thread to a session. Even if you create a global session, you should call **Session.join()** for each thread.

More specifically, API methods whose only session-implying arguments are persistent objects require that the calling thread already belong to the same session as the persistent objects. This includes global and nonglobal sessions. This restriction will be lifted in a future release. The methods affected by this restriction include the following:

- **ObjectStore.deepFetch()**
- **ObjectStore.destroy()**
- **ObjectStore.dirty()**
- **ObjectStore.evict()**
- **ObjectStore.export()**
- **ObjectStore.fetch()**
- **ObjectStore.isDestroyed()**
- **ObjectStore.isExported()**
- **ObjectStore.migrate()**
- **Persistent.deepFetch()**
- **Persistent.destroy()**
- **Persistent.dirty()**
- **Persistent.evict()**
- **Persistent.fetch()**
- **Persistent.isDestroyed()**

Use of oscompact and ossevol Utilities

Do not use the **oscompact** or **ossevol** utilities on databases created with the Java interface to ObjectStore (OSJI). Also, do not use the C++ API for these utilities on OSJI databases.

You can use the OSJI **Database.evolveSchema()** method to evolve the schema in an OSJI database. OSJI does not yet provide a compaction utility.

Notification Subscriptions Might Be Lost If Database Destroy Fails

If a call to **Database.destroy()** fails, notification subscriptions on that database might be lost. Object Design expects to fix this problem in a future release.

Hosted Pathname Syntax Might Require Setting of Environment Variable

If the directory specified in **OS_LIBDIR** uses the hosted pathname syntax (*host*:**/***dir*) and the pathname syntax for that directory has the opposite style of slash from the one for local pathnames on the client (that is, Windows and UNIX), you must set the **OS_META_SCHEMA_DB** environment variable to the pathname of the meta-schema database. That database is named **metaschm.db**, and its default location is in the **lib** subdirectory of **OS_ROOTDIR**.

Destroying Java Peer Objects

There is a bug that prevents ObjectStore from leaving a tombstone when you destroy a Java peer object. This will be fixed in a future release. For now, you must be careful that you do not destroy a Java peer object that is still referred to by another object and then try to use that reference. While doing so is always a mistake, in the current product there is no tombstone to flag the mistake for Java peer objects. When you invoke **ObjectStore.destroy()** on a primary Java object, ObjectStore leaves a tombstone. If any objects try to access the destroyed object, the tombstone causes ObjectStore to throw COM.odi.ObjectNotFoundException.

Applets

ObjectStore is a Java application that uses C++ native methods. Consequently, you cannot use ObjectStore in an applet other than through the Sun JDK Appletviewer application.

Weak References Cause Incorrect Java Garbage Collection

Persistent objects that meet both of the following conditions should be removed by the Java garbage collector:

- The persistent object was not modified.
- The application no longer refers to the persistent object.

However, when weak references are enabled, which is the default, a bug in the JDK 1.1 prevents these objects from being removed by the Java garbage collector. This problem is corrected in the beta release of the JDK 1.2.

You can work around this problem by including a call to **evict(ObjectStore.RETAIN_HOLLOW)** or **evict(ObjectStore.RETAIN_STALE)** for the relevant objects. Here is an example of the work around. For access to **OSHashtable** elements, the use of the **noCache** option to the **get()** method is very important. It prevents the **OSHashtable** instance from retaining references to the retrieved objects.

```
t = Transaction.begin(ObjectStore.UPDATE);
ht = (OSHashtable) db.getRoot("foo");
int cnt = 0;
while (++cnt > 0) {
    Object obj =
        ht.get(getKey(cnt), true);
    ... do something with obj ...
    ObjectStore.evict(obj, ObjectStore.RETAIN_HOLLOW);
}
t.commit()
```

Troubleshooting Problems — It Might Be the JIT Compiler

Object Design has tested the Just In Time (JIT) compilers available with supported and maintained platforms.

All JIT compilers tested by Object Design have exhibited one or more problems when running tests of ObjectStore. These problems often appear when NullPointerException is thrown unexpectedly, although other incorrect behavior has also been seen. Object Design has fixed the problems that have been encountered. At the time of this release, there was one outstanding problem with the Microsoft JIT that was causing a query test to get an incorrect result. You might still encounter problems in ObjectStore or in your application when you use a JIT

compiler. Object Design will continue to make every effort to fix any problems.

If you are using a JIT compiler, and you encounter a problem that you cannot diagnose, try running your application without the JIT compler. If you find that there is no problem when you run without the JIT compiler, you might be able to work around the problem and continue to run with the JIT compiler. However, you might find that you cannot use the JIT compiler.

Disabling JIT on Solaris

To disable the JDK JIT on Solaris, do one of the following:

- Set the **JAVA_COMPILER** environment variable to **NONE**:

  setenv JAVA_COMPILER NONE

- Specify **NONE** as the value of the **java.compiler** system property:

  java -Djava.compiler=NONE MyClass

Disabling Sun JDK JIT on Windows

To disable the Sun JDK JIT on Windows, do one of the following:

- Unset the **JAVA_COMPILER** environment variable:

  set JAVA_COMPILER=

- Specify the empty string for the **java.compiler** system property:

  java -Djava.compiler= MyClass

Disabling Microsoft VM JIT

To disable the JIT for the Microsoft VM, start the Internet Explorer, select Internet Options from under View, choose the Advanced tab, and make sure the Java JIT compiler enabled box is not checked.

Disabling JIT on Symantec

If you are using Symantec Visual Cafe, add the following line to the Symantec **sc.ini** file:

JAVA_COMPCMD=DISABLE;STACKTRACE

Peer Generator Incorrectly Generates Code for Some Abstract Classes

Under certain circumstances, the peer generator tool (**osjcgen**) fails to recognize that a class is abstract, and it generates code, which attempts to instantiate the class. The generated C++ code does not compile.

A work around is to suppress the generation of the methods that contain compilation errors. To do this, specify the **-suppress** option with the name of a problem method when you run the peer generator tool. This prevents the problem methods from being generated.

For example, suppose the C++ **person** class has a Java peer class generated into the **COM.people** package and there are three methods that are not compiling correctly. Run **osjcgen** and specify the **-suppress** option for each problem method.

```
-suppress COM.people.person.person \
-suppress COM.people.personU.makeArray \
-suppress COM.people.personU.set
```

The problem occurs when a C++ class inherits a pure virtual function from a base class. For example:

```
/* class A is abstract */
class A {
    public:
        virtual void f() = 0;
}

/* class B is abstract, but osjcgen treats it as nonabstract */
class B : public A {
    public:
        virtual void f() = 0;
}

/*  class C is abstract - the redeclaration of the pure virtual causes
    osjcgen to handle this correctly. */
class C : public B {
    public:
        virtual void f() = 0;
}

/* class D is nonabstract */
class D : public C {
    public:
        virtual void f();
}
```

Postprocessor Options Required for ObjectStore Collections with Indexes

If you use indexes with ObjectStore collections, you must specify the **-nothisopt** and **-noarrayopt** options when you run the postprocessor on your classes. Alternatively, you can specify the **-noinitializeropt** option in place of the two options.

This ensures that the postprocessor does not apply certain optimizations, which might cause your code to work incorrectly for evict operations performed on ObjectStore collections. These evict operations can happen during execution of the following methods:

- **addIndex()**, **query()**, **queryPick()**, and **exists()** on any collection
- **insert()**, **replaceAt()**, **insertFirst()**, **insertLast()**, **insertBefore()**, and **insertAfter()** on collections that have indexes that have **SIGNAL_DUPLICATES** behavior

Solaris: Requirement for Using Notification

The notification facility requires a native threads implementation of the JDK to work correctly on Solaris. To use the notification facility, you can use a Solaris production implementation of the JDK, which always uses native threads. Currently, the JDK 1.1.6 version is available. You can also use the separate native threads package that is sometimes available for the reference implementation of the JDK. However, the latest JDK 1.1.7 reference implementation does not currently have a native threads package available.

The standard reference implementation of the JDK release relies on green threads, which provide thread capability through Solaris asynchronous I/O and a Java internal I/O manager. When using green threads, the **Notification.receive()** method sometimes fails to return after the specified timeout has expired.

Solaris: Accessing Multithreaded C++ Applications

When OSJI applications access C++ libraries on Solaris, those libraries must be linked with **libosths** to work with the green threads version of the Java virtual machine.

The C++ interface to ObjectStore (OSC++) provides two thread libraries:

- **libosthr** ensures that calls to ObjectStore from multiple C++ threads are safe.
- **libosths** does not provide this protection.

To be used by an OSJI application running with the green threads version of the Java virtual machine, multithreaded OSC++ libraries must

- Use **libosths**.
- Use application-specific mechanisms to ensure that no two threads try to use OSC++ entry points at the same time.

If you use the native threads version of the Java virtual machine, you can link multithreaded OSC++ libraries with **libosthr** and depend on the locking primitives provided by OSC++. When you do this, the link line must include **-losthr** and either **-mt** or **-lthread**.

When you are using a Java virtual machine that is running with green threads, an attempt to use a C++ library that is linked with **libosthr** results in an error with the following message:

Application link error, please relink with the libthread library.

The message indicates that the library is multithreaded, but the Java VM is not.

Solaris: C++ Runtime Library

The OSJI installation on Solaris contains the file **libC.so.5** in its **lib** subdirectory. This file is the C++ runtime for Solaris 2.6 that is intended for use by customers who do not have C++ installed on their systems. If you are running an earlier version of Solaris and have C++ installed, you might want to ensure that your **LD_LIBRARY_PATH** is set to include the installed version of the C++ library prior to the one supplied with OSJI.

SPARCompiler 4.2 Known Problem

When you develop OSJI applications that access C++ on Solaris, you might encounter a problem when you compile the code generated by **javah**. This includes files in the JDK **include** directory. The problem produces an error message like this:

"/odi/java/JDK-1.1.7/java/include/java_lang_String.h", line 19: Error: Syntax error in pragma.

This error occurs because **javah** unconditionally produces **#pragma pack()** compiler directives that the SunSoft SC4.2 C++ compiler cannot handle. The work around is to use the SC4.0 compiler instead. This is only an issue for the Java / C++ interface.

Microsoft VM: Problems with Utility Collection Queries

The **COM.odi.util.Query** class creates new classes when optimizing queries. Because the Microsoft VM does not seem to garbage collect class objects, long running applications or applications that create many different queries might run out of heap space. Applications that use the same query object many times are not affected by this problem. The problem occurs when you

- Create and use new query objects
- Reoptimize existing query object results, which creates new classes

Problem Building Large Utility Collection Indexes

Adding an index to an existing collection (**COM.odi.util.IndexedCollection.addIndex()**) and dropping an index from a collection (**IndexedCollection.dropIndex()**) require reading all the elements of the collection in a single transaction. Doing this can result in a java.lang.OutOfMemoryError when the collection is large. If you intend the collection to be indexed, there are a few work arounds.

One work around is to increase the maximum heap space available to the Java VM. For Sun-based JDKs, you can do so with the **-mx** option.

Another work around is to add the index before inserting a large number of elements into the collection.

A third work around is to move the collection elements temporarily to a different collection before dropping an index.

This problem should be fixed in a future release.

Requirements for Using This Release

To use this release, you must satisfy requirements in the following areas.

Repostprocessing with OSJI 3.0 Postprocessor

You must use the OSJI 3.0 postprocessor to repostprocess all postprocessed class files that you used with OSJI 1.2 or 1.3.

Upgrading Databases to Use JDK 1.2

If you want to use the JDK 1.2 with OSJI 3.0, you must upgrade your database to use the correct string hash code. See Upgrading Databases for Use with the JDK 1.2 on page 107.

Additions to CLASSPATH Variable

To use ObjectStore, you must set your **CLASSPATH** environment variable to contain

- The **osji.zip** or **osji.jar** file, which contains ObjectStore
- The **tools.zip** or **tools.jar** file, which contains the**.class** files for the Class File Postprocessor and other development tools

Examples

For example, under Solaris, you might set your **CLASSPATH** variable to something like this:

```
CLASSPATH=/opt/ODI/osji/osji.zip:/opt/ODI/osji/tools.zip
```

Under Windows, you might set it to something like this:

```
CLASSPATH=c:\odi\osji\osji.zip;c:\odi\osji\tools.zip
```

Compatibility

The **tools.zip** file included in versions of OSJI prior to this release is incompatible with the **osji.zip** file for this release. To use this release, you must ensure that the **osji.zip** or **osji.jar** and **tools.zip** or **tools.jar** files distributed with this release are both in your **CLASSPATH**.

Your **CLASSPATH** can include entries for more than one Object Design Java product only if the product versions are compatible. The **osji.zip** file for OSJI 3.0 is compatible with the 3.0 **pse.zip** and **pro.zip** files.

Developing applications

To develop ObjectStore applications, you should add entries to the **CLASSPATH** variable that allow ObjectStore to find your

- Annotated (postprocessed) **.class** files
- Original unpostprocessed **.class** files

The order of these entries is important. You want ObjectStore to find the postprocessed **.class** files before it finds the unpostprocessed **.class** files. This ensures that when you run the program, the Java system finds the annotated class definitions rather than the unannotated definitions. Detailed instructions for doing this are in the *ObjectStore Java API User Guide*, Chapter 8, Automatically Generating Persistence-Capable Classes.

Running demos

To run the demos, you must add two other entries to the **CLASSPATH** variable. Instructions are in the **README.htm** file that is in the directory for each demo.

Using compatible versions

Ensure that the OSJI **lib** directory you use is from the same release as the **.zip** or **.jar** file.

Addition to PATH Variable

Set your **PATH** environment variable to contain the **bin** directory that is in the release distribution. For example, on Solaris, this might be

PATH=/usr/ucb:/usr/bin:/opt/jdk117/bin:/opt/SUNWspro/bin:/opt/ODI/OS5.1/bin**:/opt/ODI/OSJI/bin**

On Windows, it might be

PATH=c:\jdk117\bin;c:\odi\ostore\bin**;c:\odi\osji\bin**;c:\winnt\system32;c:\winnt

Native Libraries

The Java interface to ObjectStore requires C++ native libraries. The native libraries are needed by every client machine that runs an OSJI application. You can store the libraries on a central host and access them over the network. However, they must be available in the **PATH** environment variable for each client machine.

Addition on Solaris to LD_LIBRARY_PATH

On Solaris, set your **LD_LIBRARY_PATH** environment variable to contain the **lib** directory that is in the release distribution. For example:

LD_LIBRARY_PATH=/usr/lib:/opt/ODI/OS5.1/lib**:/opt/ODI/OSJI/lib**

Using the Postprocessor

The class file postprocessor is a tool for making classes persistence-capable. For simple applications, it is best to postprocess all classes together. For more complex applications, you can postprocess your classes in correctly grouped batches. For the rules for grouping classes in batches, see *ObjectStore Java API User Guide*, Chapter 8, Automatically Generating Persistence-Capable Classes, Postprocessing a Batch of Files Is Important.

Failure to postprocess the correct classes together can result in problem situations that appear when you try to run the application and that are hard to diagnose. There are postprocessor options that allow you to determine which classes are made persistence-capable.

Work around for memory limitation

The JDK 1.1 imposes a memory limitation of 16 MB unless you override it. If you receive a java.lang.OutOfMemory error during postprocessing, you must increase the run-time memory pool. Do one of the following:

- Set the **OSJCFPJAVA** environment variable to include the **-mx** option. For example, Solaris **csh** users can enter

  setenv OSJCFPJAVA "java -mx32m"

  Windows users can enter

  set OSJCFPJAVA=java -mx32m

- Edit the **osjcfp** script (Solaris) or **osjcfp.bat** script (Windows) to incorporate the **-mx** option in the invocation of Java near the end of the script. On Solaris, the line to change is

  $OSJCFPJAVA $javaargs COM.odi.filter.OSCFP $args

  On Windows, the line to change is

  %osjcfpjava% COM.odi.filter.OSCFP %1 %2 %3 %4 %5 %6 %7 %8

  Add **-mx32m** before the **COM.odi.filter.OSCFP** entry. This allows the Java virtual machine to increase the heap to 32 MB. You can increase this value further if you need to.

Mixing Platforms

The Java interface to ObjectStore is layered on top of the C++ interface to ObjectStore. Correct installation of both products is required. You must ensure that

- You install the version of each product component that is intended for the platform on which you are installing it.
- The ObjectStore Server can access the OSJI application schema databases that correspond to the client platform.

The following sections provide examples with ObjectStore Servers on Solaris and Windows platforms. However, the ObjectStore Server can be on any platform supported by ObjectStore 5.1. There are examples for

- Windows Clients with Solaris Server on page 17
- Solaris Clients with Windows Server on page 17
- Solaris and Windows Clients with Solaris Server on page 18
- Solaris and Windows Clients with Windows Server on page 19

Windows Clients with Solaris Server

For Windows clients with a Solaris server, you must

- Install the Solaris version of the ObjectStore Server on the Solaris server.
- Install the Windows version of the C++ interface to ObjectStore on each Windows client.
- Install the Windows version of the Java interface to ObjectStore on each Windows client.
- Copy the Windows version of the OSJI application schema databases to the Solaris server. The ObjectStore Server needs these for Windows clients. You must copy these files to the directory you specify when you run the **setup** command. Instructions for running the **setup** command are in the **README.htm** file in the ObjectStore installation directory.

If you do not want the Solaris server machine also to be an ObjectStore client, you do not need to install the C++ interface to ObjectStore or the Java interface to ObjectStore on the Solaris server.

Solaris Clients with Windows Server

For Solaris clients with a Windows server, you must

- Install the Windows version of the ObjectStore Server on the Windows server.
- Install the Solaris version of the C++ interface to ObjectStore on each Solaris client.
- Install the Solaris version of the Java interface to ObjectStore on each Solaris client.
- Copy the Solaris version of the OSJI application schema databases to the Windows server. The ObjectStore Server needs these for Solaris clients. You must copy these files to the directory you specify when you run the **setup** command. Instructions for running the **setup** command are in the **README.htm** file in the ObjectStore installation directory.

If you do not want the Windows server machine also to be an ObjectStore client, you do not need to install the C++ interface to ObjectStore or the Java interface to ObjectStore on the Windows server.

Solaris and Windows Clients with Solaris Server

For Solaris and Windows clients with a Solaris server, you must

- Install the Solaris version of the ObjectStore Server on the Solaris server.
- Install the Solaris version of the C++ interface to ObjectStore on each Solaris client. The Solaris server and a Solaris client can be the same machine.
- Install the Solaris version of the Java interface to ObjectStore on each Solaris client.
- Install the Windows version of the C++ interface to ObjectStore on each Windows client.
- Install the Windows version of the Java interface to ObjectStore on each Windows client.
- Copy the Windows version of the OSJI application schema databases to the Solaris server. The ObjectStore Server needs these for Windows clients. You must copy these files to the directory you specify when you run the **setup** command on Windows clients. Instructions for running the **setup** command are in the **README.htm** file in the ObjectStore installation directory.
- If you do not install OSJI on the same machine as the ObjectStore Server, you must copy the Solaris version of the OSJI application schema databases to the Solaris server. The ObjectStore Server needs these for Solaris clients. You must copy these files to the directory you specify when you run the **setup** command on Solaris clients. Instructions for running the **setup** command are in the **README.htm** file in the ObjectStore installation directory.

  If you install OSJI on the same machine as the ObjectStore Server, the Solaris OSJI application schema databases are already available to the ObjectStore Server.

  When the server has OSJI application schema databases for both Solaris and Windows, it works best if they are in separate directories. It can be confusing when both sets of files are in the **lib** subdirectory.

Solaris and Windows Clients with Windows Server

For Solaris and Windows clients with a Windows server, you must

- Install the Windows version of the ObjectStore Server on the Windows server.
- Install the Windows version of the C++ interface to ObjectStore on each Windows client. The Windows server and a Windows client can be the same machine.
- Install the Windows version of the Java interface to ObjectStore on each Windows client.
- Install the Solaris version of the C++ interface to ObjectStore on each Solaris client.
- Install the Solaris version of the Java interface to ObjectStore on each Solaris client.
- Copy the Solaris version of the OSJI application schema databases to the Windows server. The ObjectStore Server needs these for Solaris clients. You must copy these files to the directory you specify when you run the **setup** command on Solaris clients. Instructions for running the **setup** command are in the **README.htm** file in the ObjectStore installation directory.
- If you do not install OSJI on the same machine as the ObjectStore Server, you must copy the Windows version of the OSJI application schema databases to the Windows server. The ObjectStore Server needs these for Windows clients. You must copy these files to the directory you specify when you run the **setup** command on Windows clients. Instructions for running the **setup** command are in the **README.htm** file in the ObjectStore installation directory.

  If you install OSJI on the same machine as the ObjectStore Server, the Windows OSJI application schema databases are already available to the ObjectStore Server.

  When the server has OSJI application schema databases for both Solaris and Windows, it works best if they are in separate directories. It can be confusing when both sets of files are in the **lib** subdirectory.

Compatibility Between ObjectStore, PSE Pro, and PSE

You can start development of an application in PSE or PSE Pro and at a later time upgrade it to ObjectStore. This section provides information to help you plan for this transition:

API Compatibility

The API to ObjectStore is a superset of the API to PSE and PSE Pro. All features in PSE are also in ObjectStore. The only feature in PSE Pro that is not in ObjectStore is that PSE Pro can have many sessions, while ObjectStore currently allows only one session.

Database Compatibility

Databases that you create with PSE or PSE Pro cannot be accessed with ObjectStore. Databases that you create with ObjectStore cannot be accessed by PSE or PSE Pro.

However, you can copy data among PSE, PSE Pro, and ObjectStore databases. The **COM.odi.product** property allows you to do this. See *ObjectStore Java API User Guide*, Chapter 3, Using Sessions to Manage Threads, Description of COM.odi.product.

Handling of Retained Objects at Start of New Transaction is Now the Same

In previous releases of OSJI, the handling of retained objects at the start of a new transaction was different than it was for PSE / PSE Pro. PSE / PSE Pro has been modified so that it now behaves the same way as previous releases of OSJI. OSJI has not been modified in this area.

In OSJI, when you commit a transaction with **ObjectStore.RETAIN_READONLY** or **ObjectStore.RETAIN_UPDATE**, ObjectStore hollows the retained objects at the start of the next transaction. This means that even if you always commit a transaction with **ObjectStore.RETAIN_READONLY** or **ObjectStore.RETAIN_UPDATE**, between transactions you have access to the contents of only those persistent objects whose contents were read or modified in the immediately previous transaction.

In previous releases of PSE / PSE Pro, when you committed a transaction with **ObjectStore.RETAIN_READONLY** or **ObjectStore.RETAIN_UPDATE**, PSE / PSE Pro did not hollow the retained objects at the start of the next transaction. This meant that if you always committed a transaction with **ObjectStore.RETAIN_READONLY** or **ObjectStore.RETAIN_UPDATE**, between transactions you had access to the contents of all persistent objects whose contents were read or modified in the session. PSE / PSE Pro no longer works this way. It now hollows all retained objects at the start of a new transaction.

Interchangeable Postprocessed Class Files

Subject to the conditions listed below, postprocessed class files can run against PSE, PSE Pro, or ObjectStore.

Release compatibility

The releases of PSE / PSE Pro and the Java interface to ObjectStore must be compatible.

OSJI 3.0 is compatible with PSE / PSE Pro 3.0.

In the following cases, OSJI 3.0 and PSE / PSE Pro 2.x are compatible.

- You can use PSE / PSE Pro 2.*x* postprocessed files with OSJI 3.0.
- You can use OSJI 3.0 postprocessed files with PSE / PSE Pro 2.x if you specify **-compatibilitymode** when you run the postprocessor.

If you specify the **-optimizeclassinfo** option when you run the 3.0 postprocessor, the postprocessed classes are not compatible with PSE / PSE Pro 2.*x*.

Classes that you postprocess with PSE / PSE Pro 3.0 are not compatible with PSE / PSE Pro 2.*x*, unless you specify **-compatibliitymode** when you run the 3.0 postprocessor.

Background

The 3.0 postprocessor generates fewer and smaller class files. It does not generate **ClassInfo** classes for interfaces, nor does it generate **ClassInfo.createArray()** methods. The **-compatibilitymode** option to the postprocessor forces generation of postprocessed classes that are compatible with PSE / PSE Pro 2.*x*.

Common APIs

The application must use only APIs that are common to all three products.

The following APIs are included in OSJI, but not in PSE / PSE Pro:

- ObjectStore collections in **COM.odi.coll**.
- Multiple segments in a database.
- Multiple databases open at the same time in a single session.
- Java / C++ interoperability facility in **COM.odi.jcpp**.
- Objects that are larger than 16,777,211 bytes. While ObjectStore allows larger objects, PSE / PSE Pro does not.
- Multiversion Concurrency Control (MVCC).
- Segment, and database locking through the **acquireLock()** method.
- Schema evolution.
- Notification facility.
- Peer objects.
- Server restart exception classes.
- Rawfs databases.

PSE, PSE Pro, and OSJI all include the **OSDictionary**, **OSHashtable**, and **OSVector** classes in **COM.odi.util.** However, only PSE Pro and OSJI include the other classes and interfaces in the **COM.odi.util** package.

PSE Pro allows multiple simultaneous sessions. Neither PSE, nor OSJI, allow more than one session at a time.

The persistent garbage collection API is in OSJI and PSE Pro, but not in PSE.

In PSE and PSE Pro, **Transaction.checkpoint(retain)** accepts a value of **ObjectStore.RETAIN_READONLY**. OSJI does not support this value.

In OSJI and PSE Pro, the default setting for the **COM.odi.stringPoolSize** property is **"100"**. But in PSE, the default setting for this property is **"0"**.

Browsing HTML Documentation

The documentation for ObjectStore is formatted for HTML frames and also provided in PDF format that you can print. To use the documentation, you need to know the following:

Requirements for Viewing Documentation

To use HTML frames, JavaScript must be enabled.

To view ObjectStore HTML documentation, you must be using Microsoft Internet Explorer Release 3.0.2 or subsequent releases or Netscape Navigator Release 3.0 or subsequent releases. It is necessary to select the **Back** button twice to reload both text frames.

To view PDF files in Acrobat Reader, you must use Acrobat Reader 3.0 or a subsequent release.

Displaying the Documentation

To access the documentation, display the **doc/index.htm** file in your browser. The **doc** directory is in the toplevel OSJI directory. This displays the Bookshelf for ObjectStore Java Interface. It provides a selectable list of the documentation components.

Accessing PDF Files

If you want to print a hardcopy of the documentation, there are PDF versions of the books in the **osji/doc/pdf** directory. The API reference information is generated with the **javadoc** tool and a PDF version of that information is not available.

Obtaining Support

To obtain support for using this release of the Java interface to ObjectStore, send electronic mail to **support@objectdesign.com**. Each message to this address creates a support event. Someone from Technical Support responds to the message in accordance with the policies stated at the Object Design Technical Support web site.

You can access Object Design's Support Frequently Asked Questions (FAQ) search engine at **http://support.objectdesign.com/osji/faq.shtml**.

Description of Discussion List

There is a **majordomo** discussion list called **osji-discussion**. The purpose of this list is to facilitate discussion about the Java interface to ObjectStore. You are invited to share tips and comments about application design, development, and performance with other users. Employees of Object Design read the list regularly and might participate in discussions, but are not obligated to respond.

The discussion list is not an official way to report support events, support-related questions, or product enhancement requests to Object Design, Inc. For official answers to technical questions or product issues regarding the Java interface to ObjectStore, contact Object Design Technical Support at **support@objectdesign.com**.

Additional information about Object Design discussion lists is at http://support.objectdesign.com/general/majordomo.html.

Subscribing to the Discussion List

To subscribe to the discussion list, send a mail message to **majordomo@objectdesign.com**. Put the following line in the body of the message:

subscribe osji-discussion *your_email_address*

Sending Messages

To broadcast a message to the list, send mail to **osji-discussion@objectdesign.com**. To find out what commands **majordomo** accepts, send email to **majordomo@objectdesign.com** with the following in the body of the message:

help

Unsubscribing from the Discussion List

To unsubscribe from the discussion list, send a message to **majordomo@objectdesign.com**. Put the following line in the body of the message:

unsubscribe osji-discussion *your_email_address*

You receive email confirmation from **majordomo** when your message has been received. If you do not receive a confirmation after several hours, try resending the message. If you continue to receive messages from the discussion list, it might be because of one of the following reasons.

- The return address of the email message you sent is different from the email address in the body of the message. In this case, the **unsubscribe** request goes to the list administrator for approval.
- You placed **unsubscribe osji-discussion** *your_email_address* in the subject of the message instead of in the body.
- The messages were sent before you unsubscribed. If there are system problems, it can take as long as several days for a message to arrive.

Choosing Between Support and the Discussion List

When should you send mail to **support@objectdesign.com** and when should you send mail to **osji-discussion@objectdesign.com**? If you have a specific problem or question, send mail to support. If you are looking for design ideas, performance tips, or problem work arounds, send mail to the discussion list. You should not send the same request to both addresses.

Other Object Design Discussion Lists

Object Design hosts a number of discussion lists for its products. To obtain a list of these discussion groups, send mail to **majordomo@objectdesign.com**. Put the following in the body of the message:

lists

Receiving Announcements of New Releases

To receive announcements of new releases, subscribe to the **patch-info** mailing list. Send electronic mail to **majordomo@objectdesign.com** and put the following line in the body of the message:

subscribe patch-info *your_email_address your_site_ID*

If you do not know your site ID, provide your name, company, address, and phone number.

Obtaining Third-Party Persistence-Capable Classes

As a convenience, Object Design makes available a few third-party persistence-capable versions of standard Java classes.

Caution

Object Design does not provide support for these classes and does not guarantee that these classes work. While Object Design does look over these classes before providing them, Object Design is not obligated to fix them if any problems occur.

Available classes

The available files are

- **OSBigDecimal.java**
- **OSBigInteger.java**
- **OSDate.java**
- **OSGregorian.zip —** This file contains several classes, including **OSGregorianCalendar.java.**

How to get them

You can obtain these files from the Object Design FTP server, **ftp.objectdesign.com**. The user name is **jclasses**. The password is **cLa$$less**.

Disabling Microsoft VM JIT

To disable the JIT for the Microsoft VM, start the Internet Explorer, select Internet Options from under View, choose the Advanced tab, and make sure the Java JIT compiler enabled box is not checked.