ObjectStore Release Notes

ObjectStore Release 6.1 Service Pack 2 for all platforms, March 2004

ObjectStore Release Notes

ObjectStore Release 6.1 Service Pack 2 for all platforms, March 2004

© 2004 Progress Software Corporation. All rights reserved.

Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. This manual is also copyrighted and all rights are reserved. This manual may not, in whole or in part, be copied, photocopied, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from Progress Software Corporation.

The information in this manual is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear in this document.

The references in this manual to specific platforms supported are subject to change.

Allegrix, A [Stylized], ASPen, eXcelon, ObjectStore, Progress, Powered by Progress, Progress Fast Track, Progress Profiles, Partners in Progress, Partners en Progress, Progress en Partners, Progress in Progress, P.I.P., Progress Software Developers Network, Progress Results, ProVision, ProtoSpeed, SmartBeans, SpeedScript, and WebSpeed are registered trademarks of Progress Software Corporation or one of its subsidiaries or affiliates in the U.S. and/or other countries. Accelevent, Allegrix & Design, AppsAlive, AppServer, ASP-in-a-Box, BusinessEdge, Business Empowerment, Empowerment Center, Fathom, Future Proof, IntelliStream, ObjectCache, OpenEdge, PeerDirect, POSSE, POSSENET, Progress Business Empowerment, Progress Dynamics, Progress Empowerment Center, Progress Empowerment Program, Progress for Partners, Progress OpenEdge, PSEPro, PS Select, SectorAlliance, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, Technical Empowerment, Trading Accelerator, WebClient, and Who Makes Progress are trademarks or service marks of Progress Software Corporation in the U.S. and other countries.

Any other trademarks or service marks contained herein are the property of their respective owners.

ObjectStore includes the RSA Data Security, Inc. MD5 Message-Digest Algorithm. Copyright © 1991-2, RSA Data Security, Inc. Created 1991. All rights reserved.

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for specific language governing permissions and limitations under the License.
Contents

Preface .............................................................................................................. vii

Chapter 1

Release 6.1.2 Information .............................................................. 1

New and Changed Features .......................................................... 2
  Programmatic Support for Backup and Restore Operations .......... 2
  New Pathname Encoding Function ............................................... 2
  Changes to the Schema Evolution API ......................................... 3
  New Platform Support: HP-UX and AIX .................................. 4
  Multi-process Dump and Load ..................................................... 4
  Performance and Metadata Checking .......................................... 4
  Changes to the Documentation ................................................... 7

Platform and Release Compatibility .............................................. 7
  Release Compatibility ................................................................. 7
  Compiler Compatibility ............................................................... 7

Platform Configuration: Solaris 32-Bit ........................................ 8
  Platform Configuration: Solaris 64-Bit ...................................... 8

  Platform Configuration: Windows 32-Bit Visual C++ 7 .......... 10

Platform Configuration: Linux ....................................................... 10
  Platform Configuration: HP-UX 32-Bit .................................... 10
  Platform Configuration: HP-UX 64-Bit .................................... 10
  Platform Configuration: AIX ....................................................... 11

Restrictions, Limitations, and Known Problems ......................... 11
  Allocator Framework Not Supported ....................................... 11
  Incompatibilities Between Visual C++ 6 and 7 ....................... 12
Contents

Red Hat Linux 8 Address Space Limitation .......................... 13
Address-Space Release Facility ...................................... 13
Hostname Resolution and Performance ............................. 14
Generating Schema for Empty Abstract Classes on Solaris ....... 14
Least-Space Allocation Strategy .................................... 14
Fixing Incorrect Vector Headers .................................... 14
CMTL Restrictions ..................................................... 18
DDML Restrictions ....................................................... 19
Java Components of ObjectStore ..................................... 19
Using the Documentation with Netscape Browsers ................. 25

Chapter 2

Previous Release 6.1 Information ................................. 27
ossg's Default Front-End Parser .................................... 27
Improved Support for Failover ...................................... 28
Replication API ......................................................... 28
Defragmenting ObjectStore Databases ............................. 29
Support for the Sun Clusters 3.0 .................................... 29
Architecture Sets for Release 6.0 .................................... 30
  Standard Architecture Sets ......................................... 30
  Versioned Architecture Sets ....................................... 30
  User-Defined Architecture Sets ................................... 31
Neutralizing Virtual Base Classes .................................. 31
Changing the Windows Registration Location ...................... 31
Preventing Excessive Page Faulting ................................ 32
Changes to Dump and Load ......................................... 33
  Using Dump and Load to Migrate Databases ...................... 33
  New osload Features ............................................... 34
  Performance Improvements ....................................... 35
  Note on Multi-process osdump and osload ....................... 35
New Macro for Functions Used in Queries ......................... 35
Optimizing Collections .............................................. 36
Compiling Single-Threaded Applications on Solaris ............... 36
objectstore::export() Renamed .................................... 37
Changes to the ObjectStore Java Interface (OSJI) ................. 37
Preface

ObjectStore® is an object-oriented database management system suited for rapid application development and deployment in multitiered environments. It combines the data query and management capabilities of a traditional database with the flexibility and power of C++ and Java interfaces.

Purpose

This document describes changes to ObjectStore for Release 6.1 Service Pack 2.

Audience

This document is for administrators or developers responsible for the installation and maintenance of ObjectStore. It is assumed that you are familiar with the ObjectStore host platform and comfortable using the operating system.

Installing this release

For information about installing Release 6.1 Service Pack 2, see one of the following:

- ObjectStore Installation for Windows
- ObjectStore Installation for UNIX
Notation Conventions

This document uses the following conventions:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Courier</td>
<td>Courier font indicates code, syntax, file names, API names, system output, and the like.</td>
</tr>
<tr>
<td>Bold Courier</td>
<td>Bold Courier font is used to emphasize particular code.</td>
</tr>
<tr>
<td>Italic Courier</td>
<td>Italic Courier font indicates the name of an argument or variable for which you must supply a value.</td>
</tr>
<tr>
<td>Sans serif</td>
<td>Sans serif typeface indicates the names of user interface elements such as dialog boxes, buttons, and fields.</td>
</tr>
<tr>
<td>Italic serif</td>
<td>In text, italic serif typeface indicates the first use of an important term.</td>
</tr>
<tr>
<td>[ ]</td>
<td>Brackets enclose optional arguments.</td>
</tr>
<tr>
<td>{ a</td>
<td>b</td>
</tr>
<tr>
<td>...</td>
<td>Three consecutive periods indicate that you can repeat the immediately previous item. In examples, they also indicate omissions.</td>
</tr>
</tbody>
</table>

Obtaining Support

To obtain information about purchasing technical support, contact your local sales office listed at www.objectstore.net/contact (worldwide) or call 1-781-280-4833 (in the United States).

Technical Support

When you purchase technical support, the following services are available to you:

- You can send questions to support@objectstore.net. Remember to include your site ID in the body of the email message.
- You can call the Technical Support organization to get help resolving problems. If you are in North America, call 781-280-4005. If you are outside North America, refer to the Technical Support Web site.
You can file a report or question with Technical Support by going to www.objectstore.net/support.

You can access the Technical Support Web site, which includes
- A template for submitting a support request. This helps you provide the necessary details, speeding up response time.
- Frequently asked questions (FAQs) that you can browse and query.
- Online documentation for all ObjectStore products.
- White papers and short articles about using ObjectStore products.
- Sample code and examples.
- The latest versions of ObjectStore products, service packs, and publicly available patches for downloading.
- Access to a support matrix that lists platform configurations supported by this release; see www.objectstore.net/support/matrix.
- Support policies.
- Local phone numbers and hours when support personnel can be reached.

Use the ObjectStore education services site (www.objectstore.net/services) to learn about the standard course offerings and custom workshops.

If you are in North America, you can call 1-800-477-6473 x4452 to register for classes; if you are outside North America, refer to the Technical Support Web site. For information on current course offerings or pricing, send e-mail to classes@progress.com.

Web-Accessible Documentation

The www.objectstore.net/documentation Web site provides access to a full set of product documentation for the current and one previous supported release. To navigate to the documentation page, click Support for the support page, and then click Documentation. A search utility enables you to search the documents for specific information. READMEs for every Service Pack release to the present time are also available from this location. On occasion, you are likely to find additional information or documentation clarification posted between releases.
Your Comments

ObjectStore product development welcomes your comments about its documentation. Send any product feedback to support@objectstore.net. To expedite your documentation feedback, begin the subject with Doc:. For example:

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

© 2004 Progress Software Corporation. All rights reserved.

Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. This manual is also copyrighted and all rights are reserved. This manual may not, in whole or in part, be copied, photocopied, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from Progress Software Corporation.

The information in this manual is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear in this document.

The references in this manual to specific platforms supported are subject to change.

Allegrix, A [Stylized], ASPen, eXcelon, ObjectStore, Progress, Powered by Progress, Progress Fast Track, Progress Profiles, Partners in Progress, Partners en Progress, Progress en Partners, Progress in Progress, P.I.P., Progress Software Developers Network, Progress Results, ProVision, ProtoSpeed, SmartBeans, SpeedScript, and WebSpeed are registered trademarks of Progress Software Corporation or one of its subsidiaries or affiliates in the U.S. and/or other countries. Accelevent, Allegrix & Design, AppsAlive, AppServer, ASP-in-a-Box, BusinessEdge, Business Empowerment, Empowerment Center, Fathom, Future Proof, IntelliStream, ObjectCache, OpenEdge, PeerDirect, POSSE, POSSENET, Progress Business Empowerment, Progress Dynamics, Progress Empowerment Center, Progress Empowerment Program, Progress for Partners, Progress OpenEdge, PSEPro, PS Select, SectorAlliance, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, Technical Empowerment, Trading Accelerator, WebClient, and Who Makes Progress are trademarks or service marks of Progress Software Corporation in the U.S. and other countries.

Any other trademarks or service marks contained herein are the property of their respective owners.

ObjectStore includes the RSA Data Security, Inc. MD5 Message-Digest Algorithm. Copyright © 1991-2, RSA Data Security, Inc. Created 1991. All rights reserved.

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
express or implied. See the License for specific language governing permissions and limitations under the License.
Chapter 1
Release 6.1.2 Information

This release document describes Release 6.1 Service Pack 2 of ObjectStore. It includes the following sections:

- New and Changed Features on page 2
- Platform and Release Compatibility on page 7
- Restrictions, Limitations, and Known Problems on page 11

Note

Applications built with ObjectStore 6.0.x must be re-compiled and re-linked to run under 6.1.2. If the version of the compiler you used for 6.0.x differs from the version you are using for 6.1.2, you must also do the following:

- Regenerate and recompile the schema source files (using ossg’s -assf or -asof option), and regenerate the schema databases
- Migrate your ObjectStore database. For information on migrating a database, see the ObjectStore Migration Guide, which is available from Technical Support.

Note that attempting to mix and match modules compiled on different compiler versions will usually result in a compiler warning. On HP platforms, however, the compiler may not warn if you attempt to link a module compiled on aCC 3.27 with a module compiled on aCC 3.45. If the modules use virtual base classes, such mixing and matching can corrupt the database. For more information, see the Migration Guide.
New and Changed Features

The following sections describes features of ObjectStore that were added or changed for Release 6.1 Service Pack 2.

Programmatic Support for Backup and Restore Operations

The ObjectStore API has been extended for this release to include programmatic support for backup and restore operation. Using the following classes, you can write applications to perform operations similar to those you can perform with ObjectStore command-line utilities:

- os_archiver
- os_archiver_options
- os_backup
- os_backup_options
- os_recover
- os_recover_options
- os_restore
- os_restore_options

The classes and their member functions are described in the ObjectStore C++ API Reference.

New Pathname Encoding Function

A new function, objectstore::set_pathname_encoding(), has been added to the C++ API for this release. The function enables you to specify a character set encoding to override the default encoding, as specified by the platform. The signature is:

```cpp
void objectstore::set_pathname_encoding(const char* encoding);
```

where `encoding` can be one of the following:

<table>
<thead>
<tr>
<th>Valid encoding values</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII</td>
<td>7-bit ASCII</td>
</tr>
<tr>
<td>CP1252</td>
<td>Microsoft Code Page 1252 (US English)</td>
</tr>
<tr>
<td>CP932</td>
<td>Microsoft Code Page 932 (Japanese)</td>
</tr>
<tr>
<td>EUCJP</td>
<td>Extended UNIX Code (Japanese)</td>
</tr>
</tbody>
</table>
Release 6.1.1 Information

**Valid encoding values**  | **Meaning**  
--- | ---  
UTF8 | UCS Transformation Format 8  
NONE | No encoding translation; values 0x01 through 0xFF are passed through without modification.

If you supply an invalid encoding value, calling this function will result in the `err_invalid_pathname_encoding` exception.

The `objectstore::set_pathname_encoding()` function must be called before calling `objectstore::initialize()` or `objectstore::initialize_for_sessions()`.

The `objectstore::set_pathname_encoding()` function is similar to the `OS_PATHNAME_ENCODING` environment variable; for more information about the variable, see *Managing ObjectStore*, Chapter 3.

### Changes to the Schema Evolution API

The following changes have been to the schema evolution for this release:

- **New functions:**
  - `os_class_type::get_dispatch_table_pointer_offset_other_compiler()`  
  - `os_class_type::has_dispatch_table_other_compiler()`  
  - `os_object_cursor::release_address_space()`  
  - `os_schema_evolution::set_address_space_release_interval()`  
- **The addition of the `layout_only` argument to the `os_dbutil::compare_schemas()` function**  
- **New options to the following ObjectStore utilities:**
  - `osscheq -layout_only`  
  - `ossevol -address_space_release_interval`

For information about the new and changed functions, see the *ObjectStore C++ API Reference*. For information about the `osscheq` and `ossevol` utilities, see *Managing ObjectStore*. 
New Platform Support: HP-UX and AIX

Release 6.1 Service Pack 2 includes support for the HP-UX and AIX platforms. For configuration information, see the following sections in this document:

- Platform Configuration: HP-UX 32-Bit on page 10
- Platform Configuration: HP-UX 64-Bit on page 10
- Platform Configuration: AIX on page 11

For information about all platforms supported by this release, see the Support Matrix on the Technical Support web site.

Multi-process Dump and Load

Release 6.1 Service Pack 2 supports the use of multi-process dump and load on all platforms, including Windows. For more information, see Managing ObjectStore

Performance and Metadata Checking

ObjectStore performs certain checks whenever a page is fetched from the database and brought into the cache (inbound relocation) or is committed or evicted from the cache (outbound relocation). These checks involve testing the consistency between different pieces of metadata for the same page. Previously, relocation and the accompanying checking would be skipped whenever ObjectStore could determine that it was not needed for normal functional reasons.

Starting with Release 6.1 Service Pack 2, ObjectStore no longer skips relocation in the interest of additional metadata checking. For example, ObjectStore would previously skip outbound relocation for pages that do not contain pointers. This is no longer the case. Starting with this release, the default behavior is to perform relocation and checking. This default behavior allows ObjectStore to detect database corruption sooner than would be the case without the checking. When such checking occurs during inbound relocation, it can prevent an application from using a corrupt page. During outbound relocation, metadata checking can prevent database corruption from occurring in the first place.

However, the new default checking can add significant overhead in certain circumstances, especially during outbound relocation when pages that have never contained pointers are committed or evicted. As a result of the added checking, user applications may experience lowered performance in
comparison with application performance in previous releases of ObjectStore.

To enable you to turn off the added checking and restore performance to its 6.1 levels, this release of ObjectStore provides the following new, Boolean-valued environment variables:

- `OS_ALLOW_OUTBOUND_RELOC_SKIPPING`
- `OS_ALLOW_INBOUND_RELOC_SKIPPING`
- `OS_SKIP_INBOUND_VERIFY_TAGS_INDEX`
- `OS_SKIP_FREE_SPACE_CONSISTENCY_CHECK`

To disable checking, set these environment variables to true (1). The following paragraphs describe each variable:

**OS_ALLOW_OUTBOUND_RELOC_SKIPPING**
Default: false (0)

When this variable is set to true (1), modified pages that have never had pointers on them will skip outbound relocation and all associated checking - that is, no metadata checking will occur for such pages during outbound relocation. Setting this variable to true enables users to get the performance advantage of skipping outbound relocation at the expense of losing the extra checking.

**OS_ALLOW_INBOUND_RELOC_SKIPPING**
Default: false (0)

When this variable is set to true (1), pages that do not contain any data requiring inbound relocation will, under non-hetero conditions -- that is, the page architecture matches the client architecture -- skip inbound relocation. This skipping only occurs for pages that have no pointers, virtual base pointers, or virtual function table pointers. If inbound relocation is skipped, all metadata checks performed during inbound relocation will also be skipped. Setting this variable to true enables users to get the performance advantage of skipping inbound relocation at the expense of losing the extra checking.

A previous release of ObjectStore disabled inbound relocation skipping by default to allow checking for vector headers; see Fixing Incorrect Vector Headers on page 14, which describes environment variables for controlling vector-header checking. If you want to skip inbound relocation and the...
associated metadata checking, you must set these vector-header variables in addition to OS_ALLOW_INBOUND_RELOC_SKIPPING, as follows:

```
setenv OS_ALLOW_INBOUND_RELOC_SKIPPING 1
setenv OS_CHECK_VECTOR_HEADERS 0
setenv OS_FIX_VECTOR_HEADERS 0
```

If you are using the `ksh`, the command lines would be:

```
export OS_ALLOW_INBOUND_RELOC_SKIPPING=1
export OS_CHECK_VECTOR_HEADERS=0
export OS_FIX_VECTOR_HEADERS=0
```

In Windows, you can either modify the environment settings in the control panel or use the following command lines:

```
set OS_ALLOW_INBOUND_RELOC_SKIPPING=1
set OS_CHECK_VECTOR_HEADERS=0
set OS_FIX_VECTOR_HEADERS=0
```

Note that, by default, OS_CHECK_VECTOR_HEADERS is set to 1 and OS_FIX_VECTOR_HEADERS is set to 0.

**OS_SKIP_INBOUND_VERIFY_TAGS_INDEX**

Default: false (0)

When this variable is set to true (1), inbound relocation on pages that are not yet relocated in the client cache will skip the new tags index check. Setting this variable to true enables users to get the performance advantage of not checking the tags index during inbound relocation at the expense of losing the extra checking. Note that pages that skip inbound relocation will also skip the tags index check regardless of the setting of this variable. In other words, the effect of this variable is not orthogonal to the setting of the `OS_ALLOW_INBOUND_RELOC_SKIPPING` variable.

**OS_SKIP_FREE_SPACE_CONSISTENCY_CHECK**

Default: false (0)

When this variable is set to true (1), the SPFS metadata is not tested for self-consistency when a page is prepared for relocation for the first time. (ObjectStore uses SPFS metadata to control object allocation.) Note that pages for which inbound relocation is skipped will still have this check done, since the check doesn’t require any iteration over the tags. In other words, the effect of this variable is orthogonal to the setting of the `OS_ALLOW_INBOUND_RELOC_SKIPPING` variable.
Changes to the Documentation

The ObjectStore documentation has been updated for Release 6.1 Service Pack 2. Also, for documentation on migrating from earlier releases of ObjectStore to this release, see the ObjectStore Migration Guide, which is available from Technical Support.

Platform and Release Compatibility

This section lists this platforms and compilers supported by this release of ObjectStore.

The Support Matrix on the Technical Support web site (www.objectstore.net/support/matrix) contains an up-to-date list of all supported and maintained platforms. Please refer to the Support Matrix if you are in any doubt whether your compiler or operating system are supported. If your compiler is not supported, you cannot use this release of ObjectStore.

Release Compatibility

Release 6.1 Service Pack 2 of ObjectStore is not “drop-in” compatible with Release 6.0. To upgrade a 6.0.x application to run under 6.1.x, you must re-compile and re-link the application.

Compiler Compatibility

If you are using a supported compiler and are a first-time user of ObjectStore, installing ObjectStore is a straightforward process. Likewise, if you are upgrading from Release 6.0 and are using a supported compiler that is the same as the compiler you used for the previous release, the process of upgrading to Release 6.1 Service Pack 2 is straightforward.

If you are upgrading from a pre-6.0 release or your compiler has changed since the previous release, please refer to the ObjectStore Migration Guide (www.objectstore.net/documentation/migration) that is available on the Technical Support web site. The ObjectStore Migration Guide will provide detailed instructions about upgrading to Release 6.1 Service Pack 2.
Release 6.1.1 Information

Platform Configuration: Solaris 32-Bit

You can build and run 32-bit or 64-bit applications on 64-bit hardware, but you cannot build or run 64-bit applications on 32-bit hardware

Supported Operating Systems: Solaris 8, Solaris 9

Supported Clusters: Sun Cluster 3.0 5/02 for Solaris 8

Supported C++ Compilers: Sun ONE Studio 7 (C++ 5.4)

Supported Java Compilers: Sun Java 2 SDK 1.3 or 1.4

Required Patches: Solaris 8 systems require Sun Patch 108434-xx for Sun ONE Studio 7 compiler use. Solaris 9 systems require Sun Patch 111711-xx for Sun ONE Studio 7 compiler use.

Recommended Patches: Sun Patch 108528-xx is recommended but not required for all Solaris 8 systems running ObjectStore for its resolution of BugID 449415. This patch is included in the current Solaris 8 recommended patch cluster.

Note

The string xx refers to the latest available revision of the patch from Sun.

If you are using ObjectStore’s built-in failover or failover as provided by the Sun Clusters 3.0 operating system, you may be able upgrade to Release 6.1 Service Pack 2 without having to take your system out of service by performing a rolling upgrade. For more information, see ObjectStore Installation for UNIX.

Platform Configuration: Solaris 64-Bit

You can build and run 32-bit or 64-bit applications on 64-bit hardware, but you cannot build or run 64-bit applications on 32-bit hardware

Supported Operating Systems: Solaris 8, Solaris 9

ObjectStore Release Notes
Release 6.1.1 Information

**Supported Clusters**
Sun Cluster 3.0 5/02 for Solaris 8

**Supported C++ Compilers**
Sun ONE Studio 7 (C++ 5.4)

**Supported Java Compilers**
Not supported

**Required Patches**
Solaris 8 systems require Sun Patches 108434-xx and 108435-xx for C++ compiler use.
Solaris 9 systems require Sun Patch 111711-xx and 111712-xx for Sun ONE Studio 7 compiler use.

**Recommended Patches**
Sun Patch 108528-xx is recommended but not required for all Solaris 8 systems running ObjectStore for its resolution of BugID 449415. This patch is included in the current Solaris 8 recommended patch cluster.

**Unsupported Components**
OSJI
JMTL
DDML

**Note**
The string *xx* refers to the latest available revision of the patch from Sun.

If you are using ObjectStore's built-in failover or failover as provided by the Sun Clusters 3.0 operating system, you may be able upgrade to Release 6.1 Service Pack 2 without having to take your system out of service by performing a rolling upgrade. For more information, see ObjectStore Installation for UNIX.

**Platform Configuration: Windows 32-Bit Visual C++ 6**

**Supported Operating Systems**
Windows NT 4.0 SP6a
Windows 2000
Windows XP

**Supported C++ Compilers**
Microsoft Visual C++ 6.0 SP5

**Supported Java Compilers**
Sun Java 2 SDK 1.3 or 1.4
Release 6.1.1 Information

Platform Configuration: Windows 32-Bit Visual C++ 7

Supported Operating Systems
- Windows 2000
- Windows XP

Supported C++ Compilers
- Microsoft Visual Studio .NET 2002
  Note that 2003 is not supported.

Supported Java Compilers
- Sun Java 2 SDK 1.3 or 1.4

Platform Configuration: Linux

Supported Operating Systems
- Red Hat 8.0 with kernel 2.4.18 and glibc 2.293

Supported C++ Compilers
- gcc 3.2.1

Supported Java Compilers
- Sun JDK 1.3 or 1.4

Platform Configuration: HP-UX 32-Bit

Supported Operating Systems
- HP-UX 11i (11.11) 32-bit

Supported C++ Compilers
- aCC 3.45

Supported Java Compilers
- Sun JDK 1.3 or 1.4

Platform Configuration: HP-UX 64-Bit

Supported Operating Systems
- HP-UX 11i (11.11) 64-bit

Supported C++ Compilers
- aCC 3.45

Supported Java Compilers
- Not supported

Unsupported Components
- OSJI
- JMTL
- DDML
Platform Configuration: AIX

Supported Operating Systems | AIX 5.1
Supported C++ Compilers | Visual Age 6.0
Supported Java Compilers | Not supported
Unsupported Components | OSJI
                        | JMTL
                        | DDML

Restrictions, Limitations, and Known Problems

The following sections describe restrictions, limitations, and known problems when using Release 6.1 Service Pack 2. Where possible, they also describe workarounds for the problems.

For more up-to-date information about Release 6.1 Service Pack 2, see the Support Matrix on the Technical Support web site (www.objectstore.net/support/matrix).

Allocator Framework Not Supported

Release 6.1 Service Pack 2 does not support the Allocator Framework, which includes the following classes in the ObjectStore API:

- os_Allocator
- os_allocator_registry
- os_extended_allocator
- os_set_db

The documentation describing these classes has been removed from the ObjectStore C++ API Reference and the Advanced C++ API User Guide. The Allocator Framework will be supported at a future release of ObjectStore. For more information, contact Technical Support.
Incompatibilities Between Visual C++ 6 and 7

Because of changes in Microsoft’s Visual C++ compiler from version 6 (vc6) to version 7 (vc7), users who upgrade their applications from vc6 to vc7, or who wish to use both versions, may see schema incompatibilities that affect schema generation. The following sections discuss these incompatibilities and their workarounds.

Differing Support for \#pragma pack(pop,N)

Support for \#pragma pack(pop, N), where N is some alignment, differs between vc6 and vc7. In particular, the following set of pragmas results in different alignments between vc6 and vc7:

```
#pragma pack(push,11,2)
#pragma pack(push,12,4)
#pragma pack(pop,1)
```

On vc6, these pragmas result in a two-byte alignment; whereas on vc7 they result in a one-byte alignment. The workaround for this incompatibility is to replace the last pragma with either of two sets of pragmas. The first set ensures one-byte alignment that is compatible with vc7:

```
#pragma pack(pop)
#pragma pack(1)
```

The next set ensures two-byte alignment that is compatible with vc6:

```
#pragma pack(pop)
#pragma pack(2)
```

Differing Support for Integral Extension Types

On vc6, the integral extension types are distinct types. On vc7, they are treated as typedefs for regular C++ types. Thus, the following code behaves differently on vc6 and vc7:

```cpp
template<class T> class printer {
public:
    static void print(const char* source_name) {
        const type_info& ti = typeid(T);
        printf("in source %s, actual %s\n",ti.name());
    }
};
printer<_int16>::print("_int16");
printer<unsigned __int32>::print("unsigned __int32");
```

On vc6, the output is:

```
in source _int16, actual _int16
in source unsigned __int32, actual unsigned __int32
```
On vc7, the output is:

```
in source __int16, actual short
in source unsigned __int32, actual unsigned int
```

Furthermore, the symbol for `print` (or for the virtual function table, if there was one) is different. On vc6, the template argument for `__int16` would be `_F`; on vc7, it would be `F`.

If you wish to share code or databases between vc6 and vc7 applications, you cannot use integral extension types in any context that appears in the schema, including template instantiations and virtual function table pointers. If you are not sharing code between vc6 and vc7 applications, you can continue to use the integral types and the “mangling” will follow the rules for the compiler version you are using.

**Red Hat Linux 8 Address Space Limitation**

On Red Hat Linux 8, the default setting for the `vm.max_map_count` kernel parameter (65536) limits the address space to 256 MB. This can create a potential address space limitation. You can change the value of this parameter with the following command line:

```
sysctl -w vm.max_map_count= value
```

To display all kernel parameters, use the following command line:

```
sysctl -a
```

You can also work around the address space limitation by setting the ObjectStore environment variable `OS_8K_PAGE` or `OS_16K_PAGE`. Setting either of these variables has the effect of increasing the page size read by ObjectStore to 8 KB or 16 KB. In some cases, increasing the page size might result in additional lock conflicts because ObjectStore creates locks on a per-page basis. A page size of 8 KB or 16 KB can have greater chances of lock conflicts than a page size of 4 KB.

**Address-Space Release Facility**

The ObjectStore address-space release facility is not thread safe. As a result, there is the risk that one thread will release address space that is being used by another, leading to unpredictable behavior and potential database corruption. To protect against this risk, this release of CMTL disables the address-space release in software with which it is linked. It is possible that this protective measure can impose address-space limitations that might prevent a 32-bit CMTL application from being successfully deployed. A
more general solution to the problem will be implemented in a future release of CMTL.

Hostname Resolution and Performance
If ObjectStore cannot resolve a hostname (because, for example, the network was reconfigured or a database was moved to a new domain), application performance can suffer accordingly. If you think that hostname resolution is degrading the performance of your application, contact ObjectStore Technical Support.

Generating Schema for Empty Abstract Classes on Solaris
The schema generator (ossg) cannot generate schema for code that contains empty abstract classes used as virtual base classes on Solaris platforms. As a workaround, ossg will emit instructions to add a padding member.

Least-Space Allocation Strategy
You should not use the least-space allocation strategy until contacting ObjectStore Technical Support for a patch. Note that the least-space allocation strategy is not the default. The default is the least-wait allocation strategy, which you can continue to use.

For information about allocation strategies and how to set them, see objectstore::set_allocation_strategy() in Chapter 2 (“Class Library”) of the C++ API Reference.

Fixing Incorrect Vector Headers
In C++, vector headers contain data stored in front of a vector by the C++ run time. Vector headers are not normally visible to user applications. Only vectors allocated with ::operator new have vector headers; vectors embedded in objects or allocated off the stack do not have vector headers. Also, only vectors of elements that have nondefault destructors have vector headers.

The vector header encodes information that the C++ run time uses when the vector is deleted to determine how to run the destructors on the elements. If the vector header is incorrect when the vector is deleted, the C++ run time is unlikely to run the correct number of destructors on its elements. Running an incorrect number of destructors on the vector could (depending on the behavior of the destructors) result in database corruption or unpredictable application behavior.
Incorrect vector headers can occur under the following conditions:

- The vector was created on an HP-UX 11.00 (64-bit) or Solaris/SPARC 2.8 (64-bit) platform without using the normal form of the ObjectStore overloading of `::operator new`. The following is an example of the normal form:

  ```cpp
  void *p = ::operator new(
      sizeof(foo) * N, db, foo::get_os_typespec(), N);
  ```

- The vector was hetero-relocated from a different platform to an HP-UX 11.00 (64-bit), Solaris/SPARC 2.8, or Linux gcc platform; and the page containing the vector was committed.

Previous to Release 6.0 Service Pack 7, the ObjectStore client automatically fixed what were assumed to be incorrect vector headers during inbound relocation of a page only if the architecture that wrote to the page differed from the current architecture that is reading the page. Starting with Release 6.0 Service Pack 7, ObjectStore provides the following tools for checking if the vector headers are incorrect and for fixing incorrect vector headers:

- **Environment variables**: `OS_CHECK_VECTOR_HEADERS` and `OS_FIX_VECTOR_HEADERS`

- The following static functions:
  - `objectstore::set_check_vector_headers()`
  - `objectstore::get_check_vector_headers()`
  - `objectstore::set_fix_vector_headers()`
  - `objectstore::get_fix_vector_headers()`

- The `osfixvh` utility

The following sections describe how to use these tools.

**Using Environment Variables**

The following Boolean-valued environment variables can be used to check and fix incorrect vector headers:

- `OS_CHECK_VECTOR_HEADERS`
- `OS_FIX_VECTOR_HEADERS`

To enable checking or fixing of vector headers, set the appropriate variable to 1 (true). To disable checking or fixing, set the variable to 0 (false). These variables are process-global.
By default, OS_CHECK_VECTOR_HEADERS is set to 1 and OS_FIX_VECTOR_HEADERS is set to 0. In this case, the client checks all vector headers for correctness during inbound relocation and raises an exception if it finds an incorrect vector header. This behavior provides the highest degree of safety when dealing with vector headers.

If you set OS_CHECK_VECTOR_HEADERS to 0 and OS_FIX_VECTOR_HEADERS to 1, the client silently fixes all vector headers during inbound relocation, regardless of whether they are correct or not. If you set both OS_CHECK_VECTOR_HEADERS and OS_FIX_VECTOR_HEADERS to 1, the client checks all vector headers for correctness during inbound relocation, logs a message to OS_DEBUG_OUT (by default, stderr), fixes the vector headers, and continues.

There is a performance cost when either OS_CHECK_VECTOR_HEADERS or OS_FIX_VECTOR_HEADERS, or both, are set to 1. In the worst case, an application can experience up to 30% loss in performance. However, this worst case is the highly unlikely case of a database in which every object is a small 1-element vector without any pointers or vtbls, and the application does nothing except fetch pages. The more likely impact on the performance of an actual ObjectStore application is much less than 30%, perhaps well below 1%.

In any case, you should set the variables at a level of safety that best meets the needs of your configuration. For example, you might want to set OS_CHECK_VECTOR_HEADERS to 1 and OS_FIX_VECTOR_HEADERS to 0, with OS_DEBUG_OUT set to a file, so that your application stays up and running in the event of an incorrect vector header. In this case, the errors would be logged to a file so that you could determine whether your database was affected by an incorrect vector header. Alternatively, you might decide that you do not need to find out if incorrect vector headers occurred but want any such problems fixed without your intervention. In this case, you would set OS_CHECK_VECTOR_HEADERS to 0 and OS_FIX_VECTOR_HEADERS to 1.

If your configuration of platforms and usage precludes the conditions in which incorrect vector headers could occur, or if you have fixed all vector headers that were previously incorrect and have taken steps to ensure that such problems will not recur, you might decide to set both OS_CHECK_VECTOR_HEADERS and OS_FIX_VECTOR_HEADERS to 0.

If you need more information about the risk of incorrect vector headers, contact ObjectStore Technical Support.
Checking and Fixing Vector Headers Programmatically
You can check for and fix incorrect vector headers programmatically, using the following functions:

```c
static void objectstore::set_check_vector_headers(
  os_boolean value);
static os_boolean objectstore::get_check_vector_headers();
static void objectstore::set_fix_vector_headers(
  os_boolean value);
static os_boolean objectstore::get_fix_vector_headers();
```

The `set...()` forms of these functions correspond to the `OS_CHECK_VECTOR_HEADERS` and `OS_FIX_VECTOR_HEADERS` environment variables, which are described in the previous section. Passing `true` (nonzero) as the `value` argument enables checking or fixing; `false` (0) disables checking or fixing.

You can use the `get...()` forms of the functions to determine whether vector-header checking or fixing is enabled.

The considerations that apply to the use of the environment variables also apply to the functions; see Using Environment Variables on page 15.

Using `osfixvh` to Check and Fix Vector Headers
You can use the command-line utility `osfixvh` to check for and fix incorrect vector headers. To use this utility to detect and fix existing incorrect vector headers, perform the following steps:

1. Make sure that you have installed Service Pack 7 or later of ObjectStore Release 6.1 Service Pack 2.
2. To check for incorrect vector headers without fixing them, execute the following command line:
   ```bash
   osfixvh -verify_only database_name
   ```
   This command line generates a report similar to the following example:

   ```bash
   % osfixvh -verify_only foo.db
   An incorrect vector header was found during inbound relocation at 0xe1804890. The source architecture is an HP with aCC. The address corresponds to offset 0x4890 within cluster #0, segment #0 of database "/bar/foo.db".
   %
   ```

3. To fix incorrect vector headers in a database, execute the following command line:
   ```bash
   osfixvh -fix database_name
   ```
When it fixes an incorrect vector header, this command line generates a report similar to the following example:

```
% osfixvh -fix foo.db
An incorrect vector header was fixed during inbound relocation at 0xe1804890. The source architecture is an HP with aCC. The address corresponds to offset 0x4890 within cluster #0, segment #0 of database "/bar/foo.db".
%
```

The `osfixvh` utility does not automatically check for and fix incorrect vector headers in schema databases and affiliated databases. You must check and fix such databases separately. If you need assistance, please contact ObjectStore Technical Support.

**CMTL Restrictions**

The following sections describe restrictions when using the Release 6.1 Service Pack 2 version of CMTL.

**Configuring CMTL from XML on Linux Platforms**

On Linux platforms, if your application configures CMTL from an XML-based configuration stream, you must specify the `ios::in` flag when creating the `ifstream` object that is passed as an argument to `os_cache_pool_manager_configuration::create_from_xml_stream()`.

On Linux platforms, `ifstream` is not opened for reading by default. The workaround is to explicitly pass an argument to the `ifstream` constructor telling it to open the stream for reading, as in the following example:

```
ifstream xml_file(config_info, ios::in);
```

The `ios::in` flag tells the constructor to open the stream for reading. Note that adding this flag to the constructor is required only on Linux platforms.

**Setting the commit_if_idle attribute**

The CMTL cache pool attribute `commit_if_idle` value default is `true` for read-only caches (`true` is also the value for update caches). Setting the `commit_if_idle` attribute value to `false` will result in a shutdown timing problem in the CMTL virtual transaction manager thread. If you have explicitly set the `commit_if_idle` value to `false` in your cache pool configuration, you need to modify it to avoid this timing issue.
DDML Restrictions

The following restrictions apply to the Release 6.1 Service Pack 2 version of DDML:

- DDML is not supported on Solaris 64-bit (sol64) platforms.
- The following DDML warning message can be safely ignored:

```
Warning: com.odi.osdm.JosdmCPlusPlus.charPmap is a static field of type com.odi.util.OSSmallMap which might refer to a persistent object. If this field does refer to a persistent object it must be user maintained.
```

Java Components of ObjectStore

The following sections describe the restrictions concerning the Java components of ObjectStore — for example, OSJI and JMTL.

Upgrading Pre-6.0 OSJI Databases

This release does not support the use of the dump/load facility for the purpose of upgrading a pre-6.0 OSJI database to Release 6.1 Service Pack 2. If you have a pre-6.0 OSJI database that you want to upgrade, you must contact ObjectStore Technical Support.

Not Supported on 64-Bit Platforms

The Java components of ObjectStore are not supported on 64-bit platforms for this release.

Running Java Browser on UNIX

Running the Java browser on UNIX platforms from Exceed clients have the following problems:

- The Message Box is Empty, because the Windows Manager does not resize the Message Box correctly. You have to resize the Message Box so the content will appear.
- Menu position is incorrect. You need to resize the main window so the menu position is correct.

Use of JDK 1.3 and 1.4 on Solaris

To ensure compatibility between ObjectStore and JDK 1.3 or 1.4 on Solaris platforms, you must set the `LD_PRELOAD` environment variable. C shell users can set this variable with the following command line:

```
setenv LD_PRELOAD libosopdel.so
```
Bourne or Korn shell users can use the following command line:

\[ \text{LD\_PRELOAD}=\text{libosopdel\.so} \text{ ; export LD\_PRELOAD} \]

**Use of JDK 1.3 on HP-UX**

To ensure compatibility between ObjectStore and JDK 1.3 on HP-UX 32-bit platforms, you must set the \text{LD\_PRELOAD} environment variable. C shell users can set this variable with the following command line:

\[ \text{setenv LD\_PRELOAD libos\.sl:libosth\.sl} \]

Bourne or Korn shell users can use the following command line:

\[ \text{LD\_PRELOAD}=\text{libos\.sl:libosth\.sl} \text{ ; export LD\_PRELOAD} \]

**Level One Integration**

Level one integration (JTA transaction integration) is supported for BEA WebLogic Server 6.1 and BEA WebLogic Server 7.0 only.

**Terminated Sessions Do Not Release All Resources**

Currently, a terminated session does not release all of its resources until each thread in a session explicitly calls the \text{Session\.leave\(\)}, \text{Session\.terminate\(\)}, \text{Objectstore\.shutdown\(\)}, \text{Session\.create\(\)} or \text{Session\.join\(\)} method.

Terminating a session continues to make all threads leave the session for all other purposes, except for releasing the session's resources.

The \text{Session\.leave\(\)} method has been modified so it can be called by threads, even if they are not joined to a session. Previously, an exception would be thrown.

Applications must ensure that all threads explicitly leave sessions that have been terminated, so that session resources are freed. If this is not done, resources allocated to sessions are not released, resulting in a possible \text{com\_odi\_AddressSpaceFullException} being thrown when trying to create a new session.

**Threads Are Not Being Automatically Joined to Nonglobal Sessions**

Threads that do not belong to a session are not automatically joined to a nonglobal session when they should be. 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 by calling
Session.join() or using a global session.

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 applies to 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()

Schema Write-Lock Conflicts Might Occur Immediately
Following Schema Installation

Users running ObjectStore applications concurrently against a database
might encounter schema segment write-lock conflicts during a transaction
that immediately follows a transaction in which the schema was installed.

To work around this problem, run a small dummy transaction immediately
after the transaction that installed the schema. The dummy transaction must
use the database, such as looking up a database root. Using the database
validates the schema and prevents potential future write lock conflicts.
Use of ossevol Utility
Do not use the ossevol utility on databases created with the Java interface to ObjectStore (OSJI). Also, do not use the C++ API for this utility on OSJI databases.

You can use the OSJI Database.evolveSchema() method to evolve the schema in an OSJI database.

Use of osgc and oscompact Utilities
You should not run the osgc or oscompact utilities against databases that are currently opened with applications that retain references to non-exported objects. To prevent possible corruption you should close the databases before running either utility.

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 metaschema database. That database is named metaschm.db, and its default location is in the lib subdirectory of OS_ROOTDIR.

Transient Segmentation Violation Errors
On UNIX platforms there are some known interaction problems between the Java VM and OSJI. If you encounter any transient segmentation violations errors, choose a heap size and set both the initial and maximum heap size to that value on the command line. For example, if you choose a 64 MB heap size, specify both -Xms64m and -Xmx64m as arguments to the java command.

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.

Troubleshooting Problems - It Might Be the JIT Compiler
Just In Time (JIT) compilers can sometimes make the difference between acceptable and unacceptable performance. However, they can also be the source of problems you might encounter when using ObjectStore. If you experience an undiagnosed problem when using a JIT compiler on
ObjectStore, toggle the JIT compiler on and off to determine whether the JIT compiler is the source of the problem. If the problem occurs only when the JIT compiler is on, contact ObjectStore Technical Support for help in resolving the problem.

The following sections describe how to toggle off the JIT compiler to diagnose troublesome behavior.

**Disabling JIT on Solaris**

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

- Set the `JAVA_COMPILER` environment variable to `NONE`:
  ```bash
  setenv JAVA_COMPILER NONE
  ```
- Specify `NONE` as the value of the `java.compiler` system property:
  ```bash
  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:
  ```bash
  set JAVA_COMPILER=
  ```
- Specify `NONE` as the value of the `java.compiler` system property:
  ```bash
  java -Djava.compiler=NONE MyClass
  ```

**Peer Generator Incorrectly Generates Code for Certain Abstract Classes**

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

A workaround 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.

```bash
-osjcgen -suppress com.people.person.person
-osjcgen -suppress com.people.personU.makeArray
-osjcgen -suppress com.people.personU.set
```
The problem occurs when a C++ class inherits a pure virtual function from a base class. For example:

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

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

/* class C is abstract - the redeclaration of the pure virtual
    causes osjcggen 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 Peer Collections with Indexes

If you use indexes with ObjectStore peer 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 with indexes that have `SIGNAL_DUPLICATES` behavior
Accessing Multithreaded OSCI Libraries on Solaris

For information about accessing multithreaded ObjectStore C++ interface (OSCI) libraries on Solaris, see Compiling and Linking on Solaris 2 in Chapter 4 of Building ObjectStore C++ Applications. If your OSJI application is multithreaded and you want to access an OSCI library to call APIs that are not available in OSJI, contact ObjectStore Technical Support for recommendations.

Using the Documentation with Netscape Browsers

The ObjectStore documentation set is compatible with Netscape Release 4.7, 6.1, or 7.0. The documentation set cannot be properly displayed with Netscape Release 6.0 on any platform. If you only have access to Netscape Release 6.0, you should use the PDF versions of the documentation set.
Chapter 2
Previous Release 6.1
Information

This chapter describes features of ObjectStore that were documented in the release notes for the previous release of ObjectStore 6.1. This information is provided for users who are migrating to Release 6.1 Service Pack 2 and wish to know changes to ObjectStore that occurred during the intervening release.

ossg’s Default Front-End Parser

Previous to Release 6.0, the ObjectStore schema generation utility (ossg) required you to use the -edgfe option to enable an improved front-end parser that provides improved language support, including better support for nested classes and templates. Starting with Release 6.0, this parser is the default, and the -edgfe option is no longer recognized. If you wish to use the pre-Release 6.0 front-end parser, you must invoke ossg with the -auditor option. Note, however, that the pre-6.0 parser and the -auditor option will no longer be supported at the next major release of ObjectStore.

The new default front-end parser requires you to use the OS_MARK_SCHEMA_TYPE and OS_MARK_SCHEMA_TYPESPEC macros, as documented in the ObjectStore C++ API Reference, Chapter 4 (“System-Supplied Macros”), and in Building ObjectStore C++ Applications, Chapter 2 (“Working with Source Files”). For a detailed description of the ossg utility, see “ossg: Generating Schemas” in Managing ObjectStore, Chapter 4 (“Utilities”).

Note

If your makefile invokes ossg with the -edgfe option, you must edit the makefile to remove this option.
Release 6.0 provides improved support for failover, as follows:

- If you are using failover that is built into ObjectStore, you can configure failover so that the primary server and the secondary (or standby) server share the load during normal ObjectStore operations. In the event of failover, the secondary server assumes the full load.

- If you are using ObjectStore on the Sun Clusters 3.0 operating system (see Support for the Sun Clusters 3.0 on page 29), you can configure ObjectStore to use the support for failover provided by the operating system as an alternative to ObjectStore-managed failover.

For detailed information about failover, see Managing ObjectStore, Chapter 6 (“High Availability of Data”).

ObjectStore’s support for failover also allows you to perform rolling upgrades when installing a new release of ObjectStore. A rolling upgrade allows you to install a new release of ObjectStore without interrupting service to clients. For more information, see “osconfig: Configuring ObjectStore” in Managing ObjectStore, Chapter 4 (“Utilities”).

Replication API

Release 6.0 introduces the following classes for managing database replication within an application:

- os_replicator
- os_replicator_options
- os_replicator_statistic_info

For more information about these classes, see os_replicator in C++ API Reference, Chapter 2 (“Class Library”).
Defragmenting ObjectStore Databases

Release 6.0 provides support for defragmenting ObjectStore databases. This support includes:

- The `os_database::get_fragmentation()` function, which returns statistics on database fragmentation.
- The new `-f` option to the `ossize` utility. This option causes `ossize` to call the `os_database::get_fragmentation()` function.
- New server parameters that can be used to prevent fragmentation:
  - Cluster Growth Policy
  - Database File Growth Policy
  - RAWFS Partition Growth Policy
- New functions that can also be used to prevent fragmentation:
  - `os_cluster::set_size()`
  - `os_database::set_size()`
  - `os_database::set_size_in_sectors()`
- You can access these functions from the command-line with the `osdbcontrol` utility, using two new options: `-cluster_size` and `-size`

For information about defragmenting ObjectStore databases, see “Managing Database Fragmentation” in Chapter 1 (“Overview of Managing ObjectStore”) of Managing ObjectStore.

Support for the Sun Clusters 3.0

Release 6.0 supports Sun Clusters 3.0. This support enables ObjectStore to use the failover support that is built into the Sun Clusters operating system. For more information, see Improved Support for Failover on page 28.
Architecture Sets for Release 6.0

Release 6.0 supports the following categories of architecture sets for use with the `ossg -arch` option when neutralizing schema:

- Standard architecture sets
- Versioned architecture sets
- User-defined architecture sets

In addition to the above architecture sets, Release 6.0 supports two other sets for use when neutralizing schema that includes virtual base classes; see Neutralizing Virtual Base Classes on page 31.

Note also that `ossg` has a new option, `-showsets`, which lists all architecture sets and their contents.

Standard Architecture Sets

The standard architecture sets meet the needs of most applications that require neutralization. These sets are:

- `all32`
- `all64`

Note that the `all` set is no longer supported. Support for this set has been removed because `ossg` no longer supports neutralization across all 32-bit and 64-bit platforms for applications that use the collections facility. If you use the `all` set in a makefile or any other scripts, you must remove it and substitute either `all32` or `all64`.

Versioned Architecture Sets

Starting with Release 6.0, you can use a versioned architecture set. A versioned architecture set contains only those platforms that are supported on a particular release of ObjectStore. For example, `all32_610` contains only those 32-bit platforms that are supported on Release 6.1. Unlike the contents of standard architecture sets, the contents of versioned architecture sets do not change from release to release.

For more information, see “Versioned Architecture Sets” in Building ObjectStore C++ Applications, Chapter 5 (“Building Applications for Multiple Platforms”). You can list all architecture sets and their contents by invoking `ossg` with the `--showsets` option, as described in “ossg: Generating Schemas” in Managing ObjectStore, Chapter 4 (“Utilities”).
User-Defined Architecture Sets

Release 6.0 allows you to define your own architecture sets, which can be specified as arguments to ossg’s -arch option. To define an architecture set, use the OS_USER_ARCH_SET environment variable, as described in Managing ObjectStore, Chapter 3 (“Environment Variables”).

Neutralizing Virtual Base Classes

If you are neutralizing schema against all 32-bit platforms, and the schema contains virtual base classes that use other virtual base classes, the ossg schema generator for Release 6.0 will prompt you to replace the virtual base classes with forced-order base classes. Replacing virtual base classes with forced-order classes ensures that the allocation order for the virtual base classes is the same across all 32-bit platforms.

Forced-order base classes are needed because of differences in the way different platforms allocate the virtual base classes. On linux3 (gcc3) platforms, virtual base classes are allocated in inheritance order. Currently, all other 32-bit platforms use post-traversal order. This difference results in a different layout order that requires neutralization.

If you are neutralizing against all 32-bit platforms except linux3, and your schema contains virtual base classes that use virtual base classes, you can specify the all32vbtrav architecture set to prevent the need for forced-order classes during neutralization. (The use of forced-order classes can increase code size.) For more information, see “Neutralizing the Allocation Order of Virtual Base Classes” in Building ObjectStore C++ Applications, Chapter 5 (“Building Applications for Multiple Platforms”). For more information about architecture sets, see Architecture Sets for Release 6.0 on page 30.

Changing the Windows Registration Location

Release 6.0 enables you to change the Windows registry location that is used by ObjectStore. Changing the registry location is especially useful when you have embedded ObjectStore in an application. By changing the registry location, you can prevent another application that also uses ObjectStore from overwriting information in the registry location that your application uses.
Release 6.1 Information

You can use the following to change the registry location:

- `os_authentication` class — For more information about this new class, see the description of the class in Chapter 2 (“Class Library”) of the C++ API Reference.

- `osserver -r` — For more information about the new `-r` option to `osserver`, see “osserver: Starting the Server” in Chapter 4 (“Utilities”) of *Managing ObjectStore*. Note that the `oscmgr6` utility for starting the cache manager now has the same new `-r` option.

- `OS_REMOTE_AUTH_REGISTRY_LOCATION` — For more information about this new environment variable, see “osserver: Starting the Server” in Chapter 3 (“Environment Variables”) of *Managing ObjectStore*.

For information about changing the registry location on NT machines, see “Setting the Registry Location for ObjectStore (Windows Only)” in *Managing ObjectStore*, Chapter 8.

Preventing Excessive Page Faulting

Release 6.0 provides the following functions of the `objectstore` class that enable you to prevent excessive page faults by disabling address markers:

- `objectstore::get_asmarkers_useless()`  
- `objectstore::set_asmarkers_useless()`

Note, however, that if you disable address-space markers in an application for which address space consumption is critical, the application runs the risk of running out of address space.

You can also use a new environment variable, `OS_ASMARKERS_USELESS`, to disable address markers. For information about the environment variable, see the description in Chapter 3 (“Environment Variables”) of *Managing ObjectStore*. The functions are described in Chapter 2 (“Class Library”) of the C++ API Reference.
Changes to Dump and Load

Release 6.0 includes enhanced versions of the ObjectStore osdump and osload utilities. Changes to these utilities include:

- Support for multiple processors in both osdump and osload. The option to specify the use of multiprocessors is -pr.
- Support for the ability to restart the osload process.
- The mechanism for generating the source code for loader applications is moved from osdump to osload. This option to specify this operation remains -emit.
- Elimination of the -ds option to dump the database schema. The database schema is always dumped in Release 6.0.

For more information, see osdump and osload in Chapter 4 (“Utilities”) of Managing ObjectStore.

Using Dump and Load to Migrate Databases

If you plan to use osdump and osload to migrate databases from Release 5.1 or release 6.0, you should always check the ObjectStore Technical Support Web site (www.objectstore.net/support) for the most recent version of each of these utilities.

The dump and load subsystem has undergone considerable revision since the last release. Data files dumped with prior releases of osdump are not compatible with current and future releases of osload. The databases from which those files were generated will need to be re-dumped with an updated version of osdump.

Updated versions of osdump for ObjectStore 5.1 and ObjectStore 6.0 can be obtained by contacting ObjectStore Technical Support. If you need to run osdump or osload with more than one process under any version of Microsoft Windows, you need to obtain updated versions from support as well. If you need to upgrade an OSJII database on any platform you must obtain updated libraries as well.
Applications built or linked with `libosdump` or `libosload` will at the very least need to be recompiled and relinked to take advantage of recent bug fixes. The following APIs have changed as well and may require code changes for some applications:

- `os_Database_table` has changed to allocate at the cluster level rather than the segment so any function calls which used to take an `os_Segment` as an argument will now take an `os_Cluster`.
- `os_Database_table::insert()` has been replaced with explicitly named function calls. This eliminates the confusion of 12 different overloading of the `insert()` function.
- `os_Loader::load()` has been changed to `os_Loader::start_load()`.

**New osload Features**

New features of `osload` in 6.1 include:

- resumption of work in the event of failure
- an easier process to generate a schema specific `osload`
- fewer required command line arguments.

**Resumption of osload**

In the event of failure during load, do not remove any temporary databases and simply re-start `osload`. The `osload` application will look for a current work database and attempt to resume based on the state of the found work database. In general this is only useful if the failure is due to hardware or user failures, such as out of disk space, network failure, or user interruptions.

**Easier Schema Generation**

The updated process to generate a schema specific `osload`:

1. Generate a 6.1 application schema; for example, `my_new_schema.adb`
2. Emit the `osload` specializations required for that schema; for example:
   ```
   osload -emit my_new_schema.adb
   ```
3. Compile the new `osload` with the files generated in step 2. You will need to update the makefile with your application specific libraries and headers.

**Simplified osload Usage**

The usage of `osload` has changed and it no longer requires you to input the dumped file names. By default `osload` will look for a file named `db_`
table.dmp in the current directory and, using that master file, it builds the work file list. If the files are located in a directory other than the current directory the -dir switch must be used.

Performance Improvements

New features of osdump and osload in ObjectStore 6.1 and post-ObjectStore 5.1.5 include performance and scalability improvements. General performance improvements require no additional steps, while scalability improvements are achieved by running osdump or osload with more than one process. To do so use the -pr switch and provide the number of process to do the work (for example. -pr 2).

Note on Multi-process osdump and osload

There is a fair amount of overhead necessary to manage the child processes in the 6.1 release so it may at times be slower than running osload with a single process. (It is expected that this overhead will be reduced in the future). The dump and load work is partitioned among child processes by the segments in a supplied database. In general, if the segments to be dumped or loaded are large (i.e., greater than 100 MB) and you have more than one in a given database, you should use multi-process osdump or osload. On the other hand, if you have a database with 2000 segments of approximately 5 MB each or a database with one segment approximately 2GB in size, you should use osdump or osload with a single process.

New Macro for Functions Used in Queries

Release 6.0 provides a new macro, OS_MARK_QUERY_FUNCTION_WITH_NAMESPACE(), for use in schema source files to include a query function in the schema. Use this macro instead of the OS_MARK_QUERY_FUNCTION() macro when the function is a member of a class that is declared in a namespace. For more information about the OS_MARK_QUERY_FUNCTION_WITH_NAMESPACE() macro, see the C++ Collections Guide and Reference, Chapter 9.
Optimizing Collections

Previous to Release 6.0, you could optimize the creation of transient sets, lists, and cursors by manually adding an optimization flag to the constructor. The flag would instruct ObjectStore to use hard pointers instead of soft pointers for the constructed type.

Release 6.0 introduces the following changes for optimizing transient sets, lists, and cursors:

- The `os_cursor::optimized` flag is no longer needed to construct an optimized transient cursor. By default, all transient cursors use hard pointers.

- Two new defines have been added for the release: `_OS_COLL_LIST_OPTIMIZE` and `_OS_COLL_SET_OPTIMIZE`. By setting these defines in your application or on the compile line, users can globally change their applications to use transient lists and sets that are based on hard pointers, without having to specify the optimization flags in the constructors.

For more information about these optimizations, see “Compiling for Collections Optimization” in C++ Collections Guide and Reference, Chapter 2.

Compiling Single-Threaded Applications on Solaris

When using the Solaris compiler to compile single-threaded applications, it is no longer necessary to use the `-mt` option. You must use this option when compiling a Release 6.0 application only if your application uses multiple threads. Refer to the Solaris documentation for more detailed information about the `-mt` option.

Note that, if you want a Release 6.0 application to use a Release 6.0 cache manager, you must compile the application with the `-mt` option, even if it is single-threaded. In this situation, however, the performance of the client/cache interface will be the same as for a 6.0 application that uses a 6.0 cache manager — not as fast as for a 6.1 application that uses a new 6.1 cache manager. For best performance and to avoid the overhead of linking with the ObjectStore thread-safe library (`libosth`), 6.1 single-threaded applications should use the 6.1 cache manager.
Release 6.1 Information

objectstore::export() Renamed

The objectstore::export() function has been renamed objectstore::export_object() to reflect its purpose more clearly — to export persistent, top-level objects. For more information about the function, see its description in Chapter 2 (“Class Library”) of the C++ API Reference. If you wish to continue using the old name, you must define OS_EXPORT_API at the top of your source file, before the ObjectStore header files are included.

Changes to the ObjectStore Java Interface (OSJI)

Release 6.0 of the ObjectStore Java Interface (OSJI) now supports both JDK 1.3 and 1.4. Support for JDK 1.2 and earlier is no longer maintained.

Changes to JMTL

In Release 6.0, the Java Middle Tier Library (JMTL) has been incorporated into the ObjectStore product and uses the same release numbering.

com.odi.jmtl.env.JVMEnviroment Class Renamed

The com.odi.jmtl.env.JVMEnviroment class has been renamed to com.odi.env.JVMEnviroment. Java applications that use the JVMEnvironment class need to be modified and recompiled.

com.odi.jmtl.env.JVMEnviroment.initialize() Method Renamed

The com.odi.jmtl.env.JVMEnviroment.initialize() method is replaced by com.odi.env.JVMEnviroment.deploy(). Any Java classes using this method need to be modified and recompiled.
New JMTL Deployment Descriptor Format

With Release 6.0, JMTL has a new deployment descriptor format. The new deployment descriptor format requires the following XML modifications:

- The new deployment descriptor format is defined by the DTD file `jmtl-dd.dtd` located in `jmtl.jar`. To use the new format, change the `DOCTYPE` entry from:
  ```xml
  <!DOCTYPE JVMEnvironment SYSTEM "JMTL:com/odi/jmtl/xml/EnvConfig.dtd">
  ```
to:
  ```xml
  <!DOCTYPE JVMEnvironment SYSTEM "JMTL:com/odi/jmtl/jmtl-dd.dtd">
  ```

- Release 6.0 includes a DTD file `jmtl-dd-compat.dtd` that you can use to make existing JMTL 1.2 XML deployment descriptor files compatible with Release 6.0. To use this file, change the `DOCTYPE` entry from:
  ```xml
  <!DOCTYPE JVMEnvironment SYSTEM "JMTL:com/odi/jmtl/xml/EnvConfig.dtd">
  ```
to:
  ```xml
  <!DOCTYPE JVMEnvironment SYSTEM "JMTL:com/odi/jmtl/jmtl-dd-compat.dtd">
  ```

New Cache Pool Attributes

Release 6.0 includes new cache pool attributes that influence cache performance. The new attributes are:

- CachePoolManager
- CachePools
- CachePool
- CacheStorage
- Databases
- DatabaseDescriptor
- Roots
- RootDescriptor
- RootObjectDescriptor
- Component
- Method
- MethodDescriptor
- ExtentDescriptor
- FinderDescriptor

The new deployment descriptor format is defined by the DTD file `jmtl-dd.dtd` located in `jmtl.jar`. To use the new format, change the `DOCTYPE` entry from:

```xml
<!DOCTYPE JVMEnvironment SYSTEM "JMTL:com/odi/jmtl/xml/EnvConfig.dtd">
```
• CommitIfIdle
• CommitIfIdleMvcc
• GroupOpenInterval
• GroupOpenIntervalMvcc
• LockTimeout
• MaxConcurrentTransactions

For more information on these attributes, see “Declarative Configuration” in the JMTL User Guide.

Integration with WebLogic Server 7.0
Release 6.0 includes a new application server specific jar file that provides level one integration with BEA WebLogic Server 7.0.

Updated Examples Using Ant Build Files
The JMTL examples have been updated to use Apache Ant to simplify configuration, building, and deployment. For additional information on using Ant, see the README.html file in the examples directory.

Changes to the Documentation

The following changes have been made to the documentation for Release 6.0:

• The ObjectStore online documentation has been reorganized into two bookshelves:
  - C++ bookshelf
  - Java bookshelf

• The online documentation is delivered through WebHelp, which provides full-text search capability.

• All of the documentation has been consolidated into the doc directory. This directory is directly under the installation directory for ObjectStore and contains the documentation for all installable components of this release, regardless of whether or not you decide to install them all.

• The following features of ObjectStore were previously released but never documented until this release:
  - os_schema_evolution::set_explanation_level(): See the description in Chapter 2 (“Class Library”) of the C++ API Reference.
Release 6.1 Information

- `os_schema_evolution::set_resolve_ambiguous_void_pointers()`: See the description in Chapter 2 (“Class Library”) of the C++ API Reference.

- `osconfig`: See the description in Chapter 4 (“Utilities”) of Managing ObjectStore.

The JMTL User Guide contains a new chapter titled “Chapter 7: Using the JMTL Console”

Summary of Changes

The following sections summarize all user-visible changes to ObjectStore for Release 6.0, including changes that have been more fully described in the preceding sections. The information in these sections is provided as a migration aid for users who might want to know at a glance how Release 6.0 will impact their applications and scripts. Each section also includes references to the relevant parts of the ObjectStore documentation for more detailed information about the changes.

New C++ Classes and Functions

The following classes and functions have been added to the ObjectStore API for Release 6.0:

- `objectstore::get_asmarkers_useless()`
- `objectstore::set_asmarkers_useless()`
- `os_authentication`
- `os_cluster::set_size()`
- `os_database::get_fragmentation()`
- `os_database::set_size()`
- `os_database::set_size_in_sectors()`
- `os_dbutil::install_backrest_control_c_handler()`
- `os_dbutil::start_backrest_logging()`
- `os_dbutil::stop_backrest_logging()`
- `os_dbutil::svr_machine()`
- `os_replicator`
- `os_replicator_options`
Some of these functions are described in other sections of this document. For detailed information about all of the new classes and functions, see the descriptions in Chapter 2 (“Class Library”) of the C++ API Reference.

New Environment Variables

The following environment variables are new for Release 6.0:

- OS_16K_PAGE
- OS_32K_PAGE
- OS_64K_PAGE
- OS_8K_PAGE
- OS_ASMARKERS_USELESS
- OS_CORE_DIR
- OS_NETWORK_SERVICE
- OS_PREALLOCATE_CACHE_FILES
- OS_REMOTE_AUTH_REGISTRY_LOCATION
- OS_USER_ARCH_SET

Some of these environment variables are described in other sections of this document. For detailed information about all of the new environment variables, see their descriptions in Chapter 3 (“Environment Variables”) of Managing ObjectStore.

New Server Parameters

The following server parameters are new for Release 6.0:

- Cluster Growth Policy
- Database File Growth Policy
- Failover Heartbeat File
- Failover Script
- Identical Pathnames on Failover Server
- RAWFS Partition Growth Policy
- RPC Timeout
Release 6.1 Information

Some of these server parameters are described in other sections of this document. For detailed information about all of the new server parameters, see their descriptions in Chapter 2 (“Server Parameters”) of Managing ObjectStore.

New Options for ObjectStore Utilities

New options have been added to the following ObjectStore utilities:

- `osarchiv -P -T`
- `osbackup -P -r -T`
- `oscopry rawfs_dir`
- `osdbcontrol -cluster_size -size`
- `osdump -dir -pr -v`
- `osload -dir -pr -v`
- `osrecovery -s -T`
- `osreplic -c -C -P -R -s -T`
- `osrestore -s -T`
- `osserver -con -hostname -M -server_name -upgradeRAWFS`
- `oss -auditor -showsets -ignore_vbo`

Note that the `-auditor` option is provided as a migration tool and is not described in Managing ObjectStore. For information about this option, see ossg’s Default Front-End Parser on page 27.

- `ossize -f`
- `ossvrstat -databases`
- `osverifydb -all -tag`

For information about the new options refer to the descriptions of the utilities in Chapter 4 (“Utilities”) of Managing ObjectStore.

Unsupported C++ Functions

The `objectstore::export()` function is not supported and has been replaced by the `objectstore::export_object()` function. For more information, see objectstore::export() Renamed on page 37.
Unsupported Java Methods

The `JVMEnvironment.initialize()` method is not supported and has been replaced by the `JVMEnvironment.deploy()` method. For more information, see `com.odi.jmtl.env.JVMEnviroment.initialize()` Method Renamed on page 37.

Unsupported Options for ObjectStore Utilities

The following options to the `ossg` utility are no longer supported:

- The `-edgfe` option; for more information, see `ossg`’s Default Front-End Parser on page 27.
- The `-store_function_parameters (-sfp)` option. Specifying the `-store_member_functions (-smf)` option provides the same functionality; see the description of the `ossg` utility in Chapter 4 (“Utilities”) of Managing ObjectStore.
Release 6.1 Information
Index

A
abstract classes used as virtual bases 14
address markers, disabling 32
address space limitation on Linux 13
address-space release facility,
  restriction 13
all architecture set, not supported 30
all32 architecture set 30
all32vbtrav architecture set 31
all64 architecture set 30
allocation strategy, restriction 14
  -arch option (ossg) 30
architecture sets
  all 30
  all, not supported 30
  all32 30
  all32vbtrav 31
  all64 30
  standard 30
  user-defined 31
  versioned 30
attributes
  commit_if_idle 18
  -auditor option (ossg) 27
B
browsers, Netscape 25
C
  cache manager, performance 36
  changes, summarized 40
  changing the Windows registry
    location 31
classes, system-supplied
  os_authentication 31
  os_replicator 28
  os_replicator_options 28
  os_replicator_statistic_info 28
Cluster Growth Policy
  server
    parameter 29
cluster operating system
  failover 28
  support for 29
  -cluster_size option (osdbcontrol) 29
CMTL
  address-space release restriction 13
  Linux platforms 18
  commit_if_idle attribute 18
  compare_schemas()
    os_dbutil, defined by 3
cursors, optimizing 36
D
database
  defragmenting 29
  failover support 28
replimapction 28
Database File Growth Policy server parameter 29
DDML, restrictions 19
defines
   _OS_COLL_LIST_OPTIMIZE 36
   _OS_COLL_SET_OPTIMIZE 36
defragmenting databases 29
disabling address markers 32
documentation and browser support 25
documentation, changes 39
dump/load facility 33
   multi-process restriction 4
   upgrading OSJI databases 19
E
-edgfe option (ossg) 27
empty abstract classes 14
encoding for pathnames 2
environment variables
   OS_ALLOW_INBOUND_RELOC_SKIPPING 5
   OS_ALLOW_OUTBOUND_RELOC_SKIPPING 5
   OS_ASMARKERS_USELESS 32
   OS_REMOTE_AUTH_REGISTRY_LOCATION 31
   OS_SKIP_FREE_SPACE_CONSISTENCY_CHECK 6
   OS_SKIP_INBOUND_VERIFY_TAGS_INDEX 6
   OS_USER_ARCH_SET 31
err_invalid_pathname_encoding exception 3
export()
   objectstore, defined by 37
F
-f option (ossize) 29
failover support 28
forced-order base classes 31
front-end parser 27
G
generating schema, restriction 14
get_asmarkers_useless()
   objectstore, defined by 32
get_dispatch_table_pointer_offset_other_compiler()
   os_class_type, defined by 3
get_fragmentation()
   os_database, defined by 29
H
has_dispatch_table_other_compiler()
   os_class_type, defined by 3
hostname resolution 14
I
inbound relocation and performance 4
J
Java browser, problems with 19
JMTL, restrictions 19
L
level one integration, support for 20
libosth library 36
Linux platforms
   address space limitation 13
   CMTL 18
   virtual base classes 31
   lists, optimizing 36
M
macros
   OS_EXPORT_API 37
   OS_MARK_QUERY_FUNCTION_WITH_NAMESPACE 35
Index

metadata checking 4
migration aids
  - auditor option (ossg) 27
  platform and release information 7
  summary of changes 40
-mt option (Solaris compiler) 36
multithreaded applications 25
multi-threaded applications, linking 36

N
Netscape browsers, supported 25
neutralizing virtual base classes 31

O
objectstore, the class
  export() 37
  get_asmarkers_useless() 32
  set_asmarkers_useless() 32
objectstore::set_pathname_encoding() 2
optimizing
cursors 36
lists 36
sets 36
options
  -auditor (ossg) 27
  -arch (ossg) 30
  -cluster_size (osdbcontrol) 29
  -ds (osdump) 33
  -edge (ossg) 27
  -emit (osdump) 33
  -f (ossize) 29
  -mt (Solaris compiler) 36
  -pr (osdump) 33
  -pr (osload) 33
  -r (osserver) 31
  -showsets (ossg) 30
  -size (osdbcontrol) 29
OS_ALLOW_INBOUND_RELOC_SKIPPING environment variable 5
OS_ALLOW_OUTBOUND_RELOC_SKIPPING environment variable 5
OS_ASMARKERS_USELESS environment variable 32
os_authentication, the class 31
os_class_type, the class
  get_dispatch_table_pointer_offset_other_compiler() 3
  has_dispatch_table_other_compiler() 3
os_cluster, the class
  set_size() 29
_OS_COLL_LIST_OPTIMIZE define 36
_OS_COLL_SET_OPTIMIZE define 36
os_database, the class
  get_fragmentation() 29
  set_size() 29
  set_size_in_sectors() 29
os_dbutil, the class
  compare_schemas() 3
OS_EXPORT_API macro 37
OS_MARK_QUERY_FUNCTION_WITH_NAMESPACE macro 35
os_object_cursor, the class
  release_address_space() 3
OS_REMOTE_AUTH_REGISTERY_LOCATION environment variable 31
os_replicator, the class 28
os_replicator_options, the class 28
os_replicator_statistic_info, the class 28
os_schema_evolution, the class
  set_address_space_release_interval() 3
  set_explanation_level() 39
  set_resolve_ambiguous_void_pointers() 40
OS_SKIP_FREE_SPACE_CONSISTENCY_CHECK environment variable 6
OS_SKIP_INBOUND_VERIFY_TAGS_INDEX environment variable 6

Release 6.1 Service Pack 2
ObjectStore Release Notes

OS_USER_ARCH_SET environment variable 31
oscmgr6 utility 31
osconfig utility 40
osdbcontrol utility 29
osdump utility 33
OSJI, restrictions 19
osload utility 33
osscheq utility 3
osserver utility 31, 32
ossevol utility 3
ossg utility
    architecture sets 30
    empty abstract classes 14
    front-end parser 27
offsetof utility 29
outbound relocation and performance 4

P
page faults, preventing 32
pathname encoding 2
performance, using environment variables to improve 4
platform support
    AIX 11
    HP-UX 32-bit 10
    HP-UX 64-bit 10
    Linux 10
    Solaris 32-bit 8
    Solaris 64-bit 8
    Windows VC6 9
    Windows VC7 10
preventing page faults 32

Q
query functions 35

R
-r option (osserver) 31
RAWFS Partition Growth Policy server parameter 29
registry location, changing 31
release_address_space()
    os_object_cursor, defined by 3
relocation and performance 4
replicating databases 28
rolling upgrades 28

S
schema evolution, changes to 3
schema failures 12
schema generation, restriction 14
server parameters
    Cluster Growth Policy 29
    Database File Growth Policy 29
    RAWFS Partition Growth Policy 29
    set_address_space_release_interval()
        os_schema_evolution, defined by 3
    set_asmarkers_useless()
        objectstore, defined by 32
    set_explanation_level()
        os_schema_evolution, defined by 39
    set_resolve_ambiguous_void_pointers()
        os_schema_evolution, defined by 40
    set_size()
        os_cluster, defined by 29
        os_database, defined by 29
    set_size_in_sectors()
        os_database, defined by 29
sets, optimizing 36
-showsets option (ossg) 30
single-threaded applications, linking 36
-size option (osdbcontrol) 29
Solaris
    DDML restriction 19
    Solaris platforms, restriction 14
    standard architecture sets 30
    summary of changes 40

ObjectStore Release Notes
Sun Clusters 3.0 system
   failover 28
   ObjectStore support for 29

T
   threaded applications, linking 36
   thread-safe library 36

U
   upgrading OSJI databases, restriction 19
   user-defined architecture sets 31
   utilities
      oscmgr6 31
      osconfig 40
      osdbcontrol 29
      osdump 33
      osload 33
      osscheq 3
      osserver 31, 32
      ossevol 3
      ossg
         architecture sets 30
         empty abstract classes 14
         front-end parser 27
         ossize 29

V
   vector headers, problems with 14
   versioned architecture sets 30
   virtual base classes, neutralizing 31
   Visual C++
      schema incompatibilities 12

W
   Windows registry location, changing 31