Objectivity Release Notes

Version 4, Release 4.0
Objectivity Release Notes

Version 4, Release 4.0, July 14, 1997

The information in this document is subject to change without notice. Objectivity, Inc. assumes no responsibility for any errors that may appear in this document.

Copyright 1996 Objectivity, Inc. All rights reserved. This document may not be copied, photocopied, reproduced, translated or converted to any electronic or machine-readable form in whole or in part without prior written approval of Objectivity, Inc.

Objectivity and Objectivity/DB are registered trademarks of Objectivity, Inc. Objectivity/DB Fault Tolerant Option, Objectivity/FTO, Objectivity/DB Data Replication Option, Objectivity/FTO, Objectivity/DB Data Definition Language, Objectivity/DDL, Objectivity/C++ Tools.h++, Objectivity/Tools.h++, Objectivity/Smalltalk, Objectivity/SQL++, Objectivity/SQL++ ODBC Driver, and Objectivity/ODBC are trademarks of Objectivity, Inc. Other brands and products are registered trademarks or trademarks of their respective holders.

The software and information contained herein are proprietary to, and comprise valuable trade secrets of, Objectivity, Inc., which intends to preserve as trade secrets such software and information. This software is furnished pursuant to a written license agreement and may be used, copied, transmitted, and stored only in accordance with the terms of such license and with the inclusion of the above copyright notice. This software and information or any other copies thereof may not be provided or otherwise made available to any other person.

U.S. Government Restricted Rights: Use, duplication or disclosure of the software or other information by the U.S. Government or any unit or agency thereof is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and the Government is acquiring only restricted rights in the software and limited rights in any technical data provided (as such terms are defined in such clause of the DFARS). If the software or other information is supplied to any unit or agency of the U.S. other than the Department of Defense, the Government’s rights will be as defined in clause 52.227-19(c)(2) of the FAR or, in the case of NASA, in clause 18-52.227-86 (d) of the NASA Supplement to the FAR.

This document is printed on recycled paper.
## Contents

Chapter 1  **What's New and Changed?**  1-1

**General Changes**  
Software Problems and Solutions Available Online 1-1  
New and Changed Product Names 1-1  
Documentation Changes 1-3

**Objectivity/DB**  
New Features 1-4  
**oocheckls** 1-4  
**New Browser oobrowse** 1-4  
Advanced Multithreaded Server (AMS) Support 1-4  
Changed Features 1-5  
Specifying the Boot File Path 1-5  
Port Mapper Not Required 1-5  
Windows Changes 1-5

**Objectivity/DB Fault Tolerant Option**  
Changed Features 1-8  
Global Distribution of Federated Database Catalog Information 1-8  
Obsolete Member Functions 1-8  
Replaced Member Functions 1-8  
Changed Types and Member Functions 1-9  
Obsolete Tools 1-10  
Changed Tools 1-10

**Objectivity/DB Data Replication Option**  
New Features 1-10  
Changed Features 1-11  
Changed Member Functions 1-11  
Changed Tools 1-11
Objectivity/C++  1-12
New Features  1-12
  Federated Database-Wide Indexing  1-12
  Predicate Query Enhancements  1-12
  ooQuery Class  1-12
  Cursor Classes  1-12
  Specifying Memory for Large Objects  1-12
  Activating Hot Mode  1-13
  Downgrading Locks with Commit and Hold  1-13
  Obtaining the Number of Containers in a Database  1-13
  ODMG Support  1-13
Changed Features  1-13
  Specifying the Boot File Path  1-13
  ooAutostart Obsolete  1-14
  Windows Changes  1-14
Objectivity/C++ Data Definition Language  1-17
New Features  1-17
  Schema Evolution  1-17
Changed Features  1-17
  Objectivity/DDL Declares/Implements Some VArrays of Integers  1-17
  UNIX C++ Template Support Limitations  1-17
  Windows Changes  1-18
Objectivity/C++ Tools.h++  1-18
Objectivity/Smalltalk  1-18
New Features  1-18
  Objectivity/DB Data Replication Option  1-18
  Transient Instance Variables  1-18
  OoSTTransform Class  1-18
  Transaction Commit with Lock Downgrade  1-19
  Change Notification Registry for Persistent Objects  1-19
  Object Conversion  1-19
  SortedCollection Support  1-19
  OoSystemConfiguration  1-19
  Thedasafe Option Supports Multiple Smalltalk Processes  1-19
Changed Features
  Copying Persistent Objects
  "OoRefreshOpen No Longer Turns Objects into Stubs"
Objectivity/SQL++
  New Features
    GRANT and REVOKE for Stored Procedures
Objectivity/SQL++ ODBC Driver
  New Features
    32-Bit ODBC Driver Available
Online Documentation
What’s New and Changed?

This document describes new and changed features for Objectivity Version 4, Release 4.0 products on Windows, UNIX, VMS, and MacOS platforms.

General Changes

This section describes general changes to Objectivity products.

Software Problems and Solutions Available Online

For a list of open and corrected problems with Objectivity products, refer to the online release notes text file for each product you are using. You can use these files to search for specific problems or problem categories.

New and Changed Product Names

Certain products have been renamed or are new to this release. Table 1-1 summarizes the product groups for this release.

<table>
<thead>
<tr>
<th>Product Name</th>
<th>Description</th>
<th>Product Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objectivity/DB</td>
<td>Distributed object database</td>
<td>ODB</td>
</tr>
<tr>
<td>Objectivity/DB Fault Tolerant Option (Objectivity/FTO)</td>
<td>Objectivity/DB add-on that provides fault tolerance</td>
<td>FTO</td>
</tr>
<tr>
<td>Objectivity/DB Data Replication Option (Objectivity/DRO)</td>
<td>Objectivity/DB add-on that provides data replication</td>
<td>DRO</td>
</tr>
</tbody>
</table>
### Table 1-1: Objectivity Products and Their Identifiers (Continued)

<table>
<thead>
<tr>
<th>Product Name</th>
<th>Description</th>
<th>Product Identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objectivity/C++</td>
<td>C++-language binding to Objectivity/DB</td>
<td>CPP</td>
</tr>
<tr>
<td>Objectivity/C++ Data Definition Language (Objectivity/DDL)</td>
<td>C++ add-on for defining Objectivity/DB schemas</td>
<td>DDL</td>
</tr>
<tr>
<td>Objectivity/C++ Tools.h++ (Objectivity/Tools.h++)</td>
<td>C++ add-on class library integrated RogueWave Tools.h++</td>
<td>TPP</td>
</tr>
<tr>
<td>Objectivity/Smalltalk</td>
<td>Smalltalk language binding to Objectivity/DB</td>
<td>SMT</td>
</tr>
<tr>
<td>Objectivity/SQL++</td>
<td>SQL language binding to Objectivity/DB</td>
<td>SQL</td>
</tr>
<tr>
<td>Objectivity/SQL++ ODBC Driver (Objectivity/ODBC)</td>
<td>Standard Microsoft SQL API for Objectivity/SQL++</td>
<td>ODC</td>
</tr>
<tr>
<td>Online documentation</td>
<td>Documentation in FrameReader format for online viewing</td>
<td>OLD</td>
</tr>
</tbody>
</table>
## Documentation Changes

The document set has been reorganized and enhanced as shown in Table 1-2.

<table>
<thead>
<tr>
<th>Old Document</th>
<th>Change</th>
<th>New Document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guide and Windows Supplement</td>
<td>Installation and OS-specific supplemental information moved</td>
<td>Installation and Platform Notes books for each OS</td>
</tr>
<tr>
<td>Application Development Tools</td>
<td>Merged</td>
<td>Objectivity/ DB Administration</td>
</tr>
<tr>
<td>Administration</td>
<td>C++ programming interface information moved</td>
<td>Using Objectivity/ C++</td>
</tr>
<tr>
<td>Objectivity/ Partitions</td>
<td>Renamed, Objectivity/DRO added</td>
<td>Using Objectivity/ FTO and Objectivity/ DRO</td>
</tr>
<tr>
<td>C++ Application Development</td>
<td>Renamed</td>
<td>Using Objectivity/ C++</td>
</tr>
<tr>
<td>Database Development</td>
<td>Renamed</td>
<td>Using Objectivity/ C++ Data Definition Language</td>
</tr>
<tr>
<td>Getting Started with C++</td>
<td>Merged</td>
<td>Using Objectivity/ C++</td>
</tr>
<tr>
<td>Objectivity/ SQL++ Application Development</td>
<td>Split into two documents</td>
<td>Using Objectivity/ SQL ++ and Using Objectivity/ SQL ++ ODBC Driver</td>
</tr>
<tr>
<td>Release Notes</td>
<td>Software problem information moved to online files distributed with each product</td>
<td>Online release notes text files</td>
</tr>
</tbody>
</table>
Objectivity/DB

This section describes new and changed features for Objectivity/DB.

New Features

This section describes new features for Objectivity/DB.

**oocheckls**

A new tool, oocheckls, determines whether or not a lock server is running on a particular machine. For more information, see “Checking to See if the Lock Server is Running” on page 7-3 in Objectivity/DB Administration.

**New Browser oobrowse**

For Windows and MacOS platforms, there is a new browser tool, oobrowse. You can use this tool to perform queries, and browse data and types. oobrowse replaces the Tool Manager (ootoolmgr) from prior releases for these platforms.

On UNIX and VMS platforms, you can continue to use the Objectivity/DB Tool Manager, as in previous releases, to launch data browsers, type browsers, query browsers, and other administration tools. For information about oobrowse and the Tool Manager, see the “Browsing Objects and Types” chapter in Objectivity/DB Administration.

**Advanced Multithreaded Server (AMS) Support**

You can now use any supported platform as an AMS host, with the exception of MacOS.
Changed Features

This section describes changed features for Objectivity/DB.

Specifying the Boot File Path

The way you specify the boot file path was changed for administration and development tools. You can now specify the path as a host and path rather than just a network path, as follows:

```
host::path
```

This change also applies to specifying the boot file path with the `OO_FD_BOOT` environment variable for Windows and UNIX, or the `ODB$FD_BOOT` logical name for VMS. You can also use spaces in the names of files and directories in any path you specify. For more information, see the “Objectivity/DB Basics” chapter and the “Specifying Objectivity/DB Files” chapter in Objectivity/DB Administration.

Port Mapper Not Required

You no longer need a port mapper to use Objectivity/DB. The lock server now uses a default port to communicate with other processes. For information on the default port number, see Installation and Platform Notes.

Windows Changes

Win32s, Windows 3.x, Windows for Workgroups 3.x Not Supported

Objectivity/DB was built with Visual C++ 4.2, which does not support the creation of Win32s applications. Because of this change, Objectivity/DB no longer supports Win32s, Windows 3.x, Windows for Workgroups 3.x. Therefore you should ignore any reference to these systems in the printed documentation. The supported platforms are Windows 95, Windows NT 3.51 and Windows NT 4.0.

Support for Long Filenames Required

Objectivity/DB requires that the file system supports long filenames for both development and runtime environments.

Filenames for Generated Journal and Database Files

The Objectivity/DB file naming conventions for Windows are now the same as the UNIX file naming conventions. This means that by default the journal filenames and database filenames generated by Objectivity/DB will not conform to the DOS 8.3 character length restriction and will contain multiple dots.
**oobrowse Support Files**
On Windows, the following Hummingbird Communications files are no longer shipped with Objectivity/DB:

- wutil.dll
- xkeysmdb
- xlib.dll
- xaw.dll
- xm.dll
- xport.dll
- xt.dll

Instead of these files, the following files from XVT Software, Inc., are shipped with Objectivity/DB to support **ObjyTool** and **oobrowse**:

- xnmbannnn.dll
- xnmtennn.dll

where

\nnnn  Three digit release number

**Windows ooddlx Uses the temp Environment Variable**
**ooddlx** now uses the *temp* (instead of *tempdir*) environment variable to determine where to put its temporary files. (7683)

**Changes to Network Service Installation Dialog**
The Network Service Installation dialog invoked by **oosvcins.exe**, or the **Objectivity** icon in the control panel, now allows the removal of a service from the list of installed services. Once removed, the setup program for the desired release of Objectivity/DB must be run to add it. (8650)

**Windows 95 Hosts Can Use UNC Name to Specify Lock Server Directory**
You can specify the lock server system directory on a Windows 95 host by exporting the directory with the UNC name \\host\OO_LS_DIR. (7832)
**Changes to Release DLL Names**

The names of release Dynamic Link Libraries (DLLs) have been changed to a more consistent format:

\[ \text{oocnnn.dll} \]

where

- **c** Designates one of these components:
  - \( \text{db} \) = Objectivity/DB runtime
  - \( \text{chk} \) = Check for and enable options

- **nnn** Three digit release number

**Table 1-3: Changes to Release DLL Names**

<table>
<thead>
<tr>
<th>Name in Release 4.0</th>
<th>Old Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oodbnnn.dll</td>
<td>liboo380.dll</td>
<td>Objectivity/DB runtime DLL</td>
</tr>
<tr>
<td>oochknnn.dll</td>
<td>ooap380.dll</td>
<td>Objectivity/DB option-enabling DLL (enables features for licensed options in the Objectivity/DB DLL)</td>
</tr>
</tbody>
</table>

**Change to Supported NT NFS Server**

The NFS server for Windows NT supported by Objectivity is Intergraph DiskShare for Windows NT. NetManage Chameleon NFS for Windows NT is no longer supported.
Objectivity/DB Fault Tolerant Option

This section describes changed features for Objectivity/DB Fault Tolerant Option (Objectivity/FTO).

Changed Features

Objectivity/FTO (formerly Objectivity/Partitions), was changed to support Objectivity/DB Data Replication Option (Objectivity/DRO). See Using Objectivity/FTO and Objectivity/DRO for more information.

Global Distribution of Federated Database Catalog Information

To support Objectivity/DRO, the federated database catalog information that was found only in the federated database autonomous partition, is now distributed to all partitions. Distributing this information provides faster access to objects outside your application’s boot autonomous partition.

Obsolete Member Functions

Due to changes to support Objectivity/DRO, the following member functions are obsolete:

ooRefHandle (ooFDObj)::controlledBy
ooRefHandle (ooDBObj)::returnControl

Replaced Member Functions

To support Objectivity/DRO, the following member functions have been replaced by new ones.

ooRefHandle (ooDBObj)::controlledBy
  ooRefHandle (ooDBObj)::controlledBy was replaced by
  ooRefHandle (ooDBObj)::containingPartition.

ooRefHandle (ooDBObj)::transferControl
  ooRefHandle (ooDBObj)::transferControl was replaced by
  ooRefHandle (ooDBObj)::changePartition.
**ooRefHandle (ooAPObj)::controls**

Was split into two member functions, `ooRefHandle (ooAPObj)::imagesContainedIn` and `ooRefHandle (ooAPObj)::containersControlledBy`.

### Changed Types and Member Functions

**ooContainsFilter**

The constant `oocInBootAp` has been replaced by `oocNotTransferred`. The constant `oocNotInBootAP` has been replaced by `oocTransferred`.

**ooRefHandle (ooFDObj)::contains**

Previously, `ooRefHandle (ooFDObj)::contains` no longer accepts a filter argument. It now does what it used to do with the default filter argument.

**ooItr (ooDBObj)::scan**

Previously, `ooItr (ooDBObj)::scan` no longer accepts a filter argument. It now does what it used to do with the default filter argument.

**ooRefHandle (ooContObj)::controlledBy**

Previously, `ooRefHandle (ooContObj)::controlledBy` previously returned the controlling partition of its containing database even if the container had not been transferred. Now, if control of the container has been transferred to a partition, this member function returns that partition; otherwise, it returns null.

**ooDBObj::operator new**

Previously, if no partition was specified, `ooDBObj::operator new` created the new database in the global partition. Now, if no partition is specified, the operator creates the database in the boot partition.

**ooDBObj::ooDBObj**

The default constructor for databases, `ooDBObj::ooDBObj`, has a new argument, `weight`, used to set the weight of the initial image of the database.

**ooRefHandle (ooAPObj)::returnAll**

Previously, `ooRefHandle (ooAPObj)::returnAll` now returns only the containers controlled by that autonomous partition to their database. The database images controlled by that autonomous partition remain where they are.
**ooDelete**
It is now an error to attempt to delete an autonomous partition that contains the last image of a database.

**Obsolete Tools**
**oochangeap**
The functions that oochangeap performed in earlier releases have been transferred to oochange.

**Changed Tools**
**oochange**
oochange now changes autonomous partition attributes as well as federated database attributes.

**oonewdb**
oonewdb now creates the database in the boot partition instead of the federated database partition in Objectivity/FTO environments.

**Objectivity/DB Data Replication Option**
This section describes new and changed features for Objectivity/DB Data Replication Option (Objectivity/DRO).

**New Features**
With Objectivity/DRO you can replicate databases on any host anywhere in your network. Objectivity/DRO improves the availability of critical data, and enhances read performance.

For more information, see the “Objectivity/DB Data Replication Option Basics” chapter in Using Objectivity/FTO and Objectivity/DRO.
**Changed Features**

**Changed Member Functions**

**Reported Errors if Multiple Database Images Exist**
The following functions return errors if more than one image of the specified database exists. They work as expected for single-image databases:

```cpp
ooRefHandle(ooDBObj)::fileName
ooRefHandle(ooDBObj)::pathName
ooRefHandle(ooDBObj)::hostName
ooRefHandle(ooDBObj)::change
```

The equivalent member functions for multiple-image databases are as follows:

```cpp
ooRefHandle(ooDBObj)::getImageFileName
ooRefHandle(ooDBObj)::getImagePathName
ooRefHandle(ooDBObj)::getImageHostName
ooRefHandle(ooDBObj)::changePartition
```

See the “C++ Interface” appendix in *Using Objectivity/FTO and Objectivity/DRO*.

**Changed Tools**

**oodumpcatalog**

`oodumpcatalog` now prints out information for database images if you are using Objectivity/FTO.

**oonewdb**

`oonewdb` now creates the database in the boot partition instead of the federated database partition in Objectivity/FTO environments. It now takes an optional argument, `-weight`, which assigns a weight to the initial image of a database.

**oochangedb**

The `-ap` flag for `oochangedb` no longer specifies the partition to which a database image will be moved. The new `-moveap` flag performs this function. The `-ap` flag now designates which autonomous partition to modify. The `-weight` flag is also new. This flag changes the weight of a database image. All other flags now apply to the designated database images rather than to the database itself.
Objectivity/C++

This section describes new and changed features for Objectivity/C++.

New Features

This section describes new features for Objectivity/C++.

Federated Database-Wide Indexing
With Objectivity/C++, you can create object indexes on a container, across databases, or across the entire federated database. For more information, see the “Using Indexes” chapter in Using Objectivity/C++.

Predicate Query Enhancements
Objectivity/C++ provides enhancements to the predicate query feature to support using indexes for certain kinds of queries. It also was enhanced to allow to-one associations as member fields. For more information, see “Using Predicate Query” on page 9-18 in Using Objectivity/C++.

ooQuery Class
Objectivity/C++ provides a new ooQuery class that you can use with the predicate query feature to iterate over sequences other than those defined by associations and containment. For more information, see “Using ooQuery” on page 9-24 in Using Objectivity/C++.

Cursor Classes
Objectivity/C++ provides a new cursor class for use with GUI applications that use scrollable lists. For more information, see the “Using the Cursor Class” chapter in Using Objectivity/C++.

Specifying Memory for Large Objects
With Objectivity/C++, you can specify the maximum amount of memory that can be used for large objects (for example, objects larger than a page). For more information, see the “Transaction Model” chapter in Using Objectivity/C++.
Activating Hot Mode

Objectivity/C++ provides a hot mode that improves performance by postponing certain object-close operations until the last possible moment. Hot mode is useful for applications that are data intensive and access many objects that all fit in memory. For more information, see the “Transaction Model” chapter in Using Objectivity/ C++.

Downgrading Locks with Commit and Hold

Objectivity/C++ provides a new parameter to the ooTrans::commitAndHold member function that you can use to downgrade all locks held by the transaction to read locks. For more information, see the “Transaction Model” chapter in Using Objectivity/ C++.

Obtaining the Number of Containers in a Database

You can use the new ooRefHandle (ooDBObj)::numContObjs member function to obtain the number of containers in a database. For more information, see the “Using Informational Functions” chapter in Using Objectivity/ C++.

ODMG Support

Objectivity/C++ provides types and classes that conform to a subset of Release 1.1 of the Object Database Management Group (ODMG) interface. For more information, see the “Using the ODMG Interface” chapter in Using Objectivity/ C++.

Changed Features

This section describes changed features for Objectivity/C++.

Specifying the Boot File Path

The way you specify the boot file path was changed for the member function ooRefHandle (ooFDObj)::open. You can now specify the path as a host and path rather than just a network path. For more information, see “Opening a Federated Database” on page 5-3 in Using Objectivity/ C++.
ooAutostart Obsolete

The ooAutostart feature is obsolete in Objectivity/C++, and will be removed from the product in the next Objectivity/C++ release. ooAutostart was designed for simple, single-database, single-transaction prototype applications. It automated certain initialization and shutdown activities for Objectivity/DB.

Windows Changes

Visual C++ 4.0 and 4.1 Not Supported
Objectivity/DB does not support the use of Visual C++ 4.0 or 4.1 for development. You must use Visual C++ 4.2.

Changes to Debug DLL Names
The names of the debug DLL has been changed to a more consistent format:

`oodbnnd.dll`

where

`db` Designates the Objectivity/DB runtime DLL
`nnn` Three digit release number
`d` Designates the debug version. Not used for release version of DLL.

<table>
<thead>
<tr>
<th>Name in Release 4.0</th>
<th>Old Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>oodbnnd.dll</code></td>
<td><code>oom380d.dll</code></td>
<td>Compatible with <code>-MDd</code> compiler option</td>
</tr>
</tbody>
</table>
Long Header Filenames
Objectivity/C++ uses long header filenames instead of DOS 8.3 filenames. (You may continue to use short include names, but we do not recommend it. While this release allows use of short include names, future releases will not.) The mapping between these long filenames and the DOS 8.3 filenames used in prior releases are shown in Table 1-5:

<table>
<thead>
<tr>
<th>Version 4 Long Filenames</th>
<th>filenames in Previous Releases</th>
</tr>
</thead>
<tbody>
<tr>
<td>ooRecover.h</td>
<td>oorecovr.h</td>
</tr>
<tr>
<td>ooSQLtypes.h</td>
<td>ooSQLtyp.h</td>
</tr>
<tr>
<td>ooMachine.h</td>
<td>oomachin.h</td>
</tr>
<tr>
<td>ooCompiler.h</td>
<td>oocmpile.h</td>
</tr>
<tr>
<td>ooPredef_ref.h</td>
<td>oopredfr.h</td>
</tr>
<tr>
<td>ooMap_ref.h</td>
<td>oomapref.h</td>
</tr>
<tr>
<td>ooUserOper.h</td>
<td>oousropr.h</td>
</tr>
<tr>
<td>ooIndex_ref.h</td>
<td>ooindxrf.h</td>
</tr>
</tbody>
</table>

Changes to Library and Static Link Object Filenames
The administration API library has been combined with the Objectivity runtime library. The names of libraries have also been changed to a more consistent format. This change does not affect users who use Objectivity/C++ automatic linking.

```
  ooc[i][d].lib
  ooc[i][d].obj
```

Where

- **c** Designates one of these components:
  - db = Objectivity runtime
  - ft = Objectivity/FTO
  - dr = Objectivity/DRO

- **i** Designates an import library

- **d** Designates the debug version when set to d. Not used for release version of libraries.
Diagnostic Memory Tracking Feature
Diagnostic memory tracking allows you to use Objectivity/C++ with memory tracking tools such as the Visual C++ debug runtime library and bounds checker to find memory leaks in your application code. (6435)

Debug ooprint Function
When you build a debug project (link with the Objectivity debug runtime library), the ooprint function can be called from within the Visual C++ debugger to view the contents of an object. (7808)

Sample Programs Modified
By default, the sample programs now build a standalone (no lock server required) application. The project files also build the DDL files automatically.

Table 1-6: Changes to Library and Static Link Object Filenames

<table>
<thead>
<tr>
<th>Name in Release 4.0</th>
<th>Old Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oodb.lib</td>
<td>liboo.lib</td>
<td>Objectivity/DB static release library</td>
</tr>
<tr>
<td>oodbdb.lib</td>
<td>N/A</td>
<td>Objectivity/DB static debug library</td>
</tr>
<tr>
<td>oodblli.lib</td>
<td>liboodll.lib</td>
<td>Objectivity/DB dynamic import release library</td>
</tr>
<tr>
<td>oodbid.lib</td>
<td>oomfcd.lib</td>
<td>Objectivity/DB dynamic import debug library</td>
</tr>
<tr>
<td>ooft.obj</td>
<td>oopart.obj</td>
<td>Objectivity/FTO static link object</td>
</tr>
<tr>
<td>oodr.obj</td>
<td>N/A</td>
<td>Objectivity/DRO static link object</td>
</tr>
<tr>
<td>N/A</td>
<td>libooadm.lib</td>
<td>Static administration library (merged into Objectivity/DB libraries)</td>
</tr>
<tr>
<td>N/A</td>
<td>ooadmdll.lib</td>
<td>Dynamic administration library (merged into oodbi.lib)</td>
</tr>
</tbody>
</table>
Objectivity/C++ Data Definition Language

This section describes new and changed features for Objectivity/C++ Data Definition Language (Objectivity/DDL).

New Features

This section describes new features for Objectivity/DDL.

Schema Evolution

Objectivity/DDL supports schema evolution, enabling you to evolve a data model as the needs of your applications and users change, and convert existing data using the modified data model. Through schema evolution, you can make changes to existing classes, their contents, and their inheritance. For more information, see the “Schema Evolution” chapter in Using Objectivity/C++ Data Definition Language and the “Object Conversion” chapter in Using Objectivity/C++.

Changed Features

This section describes changed features for Objectivity/DDL.

Objectivity/DDL Declares/Implements Some VArrays of Integers

Objectivity/DDL implements and declares the types ooVArray(uint32), ooTArray(uint32), and ooTArray(char). Objectivity/DDL already implements and declares ooVArray(char). You should remove any implement or declare statements for these types from existing schema or application code. Alternatively, you can #ifndef protect any primitive type ooVArrays you declare. For more information, see the “Data Definition Language” chapter in Using Objectivity/C++ Data Definition Language.

UNIX C++ Template Support Limitations

Objectivity/C++ supports persistent template classes wherever possible. However, support for persistent template classes is disabled when you use a C++ compiler that has no implementation, or only a partial one, for template classes. For a list of platforms and compilers that do not support persistent template classes, refer to the online release notes text file.
Windows Changes

Building DDL Files Automatically from a Visual C++ Project File
You can now use the Visual C++ IDE Custom Build rules to run Objectivity/DDL automatically when DDL source files have been updated.

Objectivity/C++ Tools.h++
No new or changed features for Objectivity/C++ Tools.h++.

Objectivity/Smalltalk
This section describes new and changed features for Objectivity/Smalltalk.

New Features
This section describes new features for Objectivity/Smalltalk. For more information on these features, see Using Objectivity/Smalltalk.

Objectivity/DB Data Replication Option
Objectivity/Smalltalk supports Objectivity/DRO.

Transient Instance Variables
Objectivity/Smalltalk now supports an instance variable type ooTransient. If you type an instance variable as <ooTransient>, the persistence mechanism ignores the variable; it does not add the variable to the schema, and it does not touch the variable when an object changes to and from being a stub.

OoSTTransform Class
OoSTTransform is an abstract class whose subclasses serve as a mechanism to store an object in a form different from the in-image representation. This framework provides a way to indicate that all instances of a particular class are to be transformed for storage.
**Transaction Commit with Lock Downgrade**

OoSession now supports the ability to commit the changes made in a transaction, retain all locks, and downgrade all write locks to read locks. To use this feature, use `OoSession>>commitAndDowngradeTransaction` in place of `OoSession>>commitTransaction`.

**Change Notification Registry for Persistent Objects**

The `OoChangeNotificationRegistry` class provides a registry to support change notification for persistent objects. This class supports two kinds of change notification: the standard `#change` mechanism, and a mechanism that allows you to send any unary message to an object when another user commits a change to the container holding the registered object.

**Object Conversion**

Objectivity/Smalltalk automatically converts an object from an older class layout to the current layout as it reads the object from the database. If you want to give a class special handling during conversion, you can implement the method `ooConvert:aConversionObject` on any class to remap old information to new instance variables as needed.

**SortedCollection Support**

Objectivity/Smalltalk now supports `SortedCollections` as persistent-capable if they use the default Smalltalk `sortBlock`.

**OoSystemConfiguration**

The `OoSystemConfiguration` class holds default value settings for the system. You can set or read the following options using standard accessor methods on the class: `defaultInitialPages`, `defaultMaxPages`, `defaultOpenMode`, `rootsClass`, and `openOnInspect`.

**Threading Option Supports Multiple Smalltalk Processes**

Two subclasses of `OoSession—OoSafeGCProtectedSession` and `OoSafeSession` enable multiple Smalltalk processes to share a single database session safely. You can use both safe and standard sessions simultaneously in a single image.
Changed Features

This section describes changed features for Objectivity/Smalltalk.

Copying Persistent Objects

A simple Smalltalk copy message sent to a persistent object now returns a non-persistent instance of the user class. Use `OoSession>>ooCopy:into:` to make a persistent copy of a persistent object. This message creates a new persistent object with its own OID.

OoRefreshOpen No Longer Turns Objects into Stubs

`OoContainer>>OoRefreshOpen`, which you can use in an MROW transaction to get the current state of an object, no longer turns in-image persistent objects into stubs. Instead, it immediately rereads the objects from the database.

Objectivity/SQL++

This section describes new features for Objectivity/SQL++.

New Features

**GRANT and REVOKE for Stored Procedures**

GRANT and REVOKE privileges are now available for stored procedures. To use a stored procedure, the database administrator must grant you privileges to execute them. For more information, see the “Using Stored Procedures” chapter in Using Objectivity/SQL++.
Objectivity/SQL++ ODBC Driver

This section describes new features for Objectivity/SQL++ ODBC Driver (Objectivity/ODBC).

New Features

32-Bit ODBC Driver Available

In addition to the 16-bit ODBC driver, Objectivity/ODBC now provides a 32-bit version.

Online Documentation

The online documentation was updated to reflect the new and changed features for this release. On Windows platforms, Objectivity/C++API reference information is available through Windows Help.