



Objectivity Release Notes

Release 10.1.2

Objectivity Release Notes

Part Number: 10.1.2-RN-0

Release 10.1.2, March 24, 2011

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 1993–2011 by 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. Active Schema, Objectivity/DB Active Schema, Assist, Objectivity/Assist, ooAssistant, Objectivity/DB ooAssistant, Objectivity/DB Fault Tolerant Option, Objectivity/FTO, Objectivity/DB Data Replication Option, Objectivity/DRO, Objectivity/DB High Availability, Objectivity/HA, Objectivity/DB Hot Failover, Objectivity/DB In-Process Lock Server, Objectivity/IPLS, Objectivity/DB Open File System, Objectivity/OFS, Objectivity/DB Parallel Query Engine, Objectivity/PQE, Objectivity/DB Persistence Designer, Objectivity/DB Secure Framework, Objectivity/Secure, Objectivity/C++, Objectivity/C++ Data Definition Language, Objectivity/DDL, Objectivity/Dashboard, Objectivity/C++ Active Schema, Objectivity/C++ Standard Template Library, Objectivity/C++ STL, Objectivity/C++ Spatial Index Framework, Objectivity/Spatial, Objectivity for Java, Objectivity/.NET, Objectivity/.NET for C#, Objectivity/Python, Objectivity/Smalltalk, Objectivity/SQL++, Objectivity/SQL++ ODBC Driver, Objectivity/ODBC, Objectivity Event Notification Services, and Persistence Designer are trademarks of Objectivity, Inc.

Other trademarks and products are the property of their respective owners.

ODMG information in this document is based in whole or in part on material from *The Object Database Standard: ODMG 2.0*, edited by R.G.G. Cattell, and is reprinted with permission of Morgan Kaufmann Publishers. Copyright 1997 by Morgan Kaufmann Publishers.

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.

RESTRICTED RIGHTS NOTICE: Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the Objectivity, Inc. license agreement and as provided in DFARS 227.7202-1(a) and 227.7202-3(a) (1998), and FAR 12.212, as applicable. Objectivity, Inc., 640 West California Avenue, Suite 210, Sunnyvale, CA 94086-3624.

Contents

| | |
|---|-----------|
| Getting Help | 5 |
| How to Reach Objectivity Customer Support | 5 |
| Before You Call | 5 |
| Chapter 1 Release Overview | 7 |
| New Products and Features | 8 |
| Updated Products | 8 |
| Products With New, Changed, or Removed Features | 8 |
| Products Rebuilt for Compatibility | 8 |
| New Platforms | 9 |
| New and Updated Books | 10 |
| Books in PDF | 10 |
| Books in HTML | 11 |
| Books in Compiled HTML Help | 12 |
| Advanced Topics | 12 |
| Chapter 2 New and Changed Features | 13 |
| Objectivity/DB (Database Services) | 13 |
| Objectivity/DB (Tools) | 15 |
| Objectivity/C++ | 16 |
| Objectivity for Java | 16 |
| Objectivity/DB Active Schema for Java | 17 |
| Objectivity/.NET for C# | 18 |
| Objectivity/Python | 19 |

| | | |
|------------------|--|-----------|
| Chapter 3 | Release Compatibility | 21 |
| | Upgrading Existing Federated Databases | 21 |
| | Selecting the Upgrade Tasks to Perform | 22 |
| | Updating to an Objectivity Release 10 License | 23 |
| | Upgrading to the Release 9.0 Internal Database Format | 24 |
| | Upgrading the Schema to Release 9.3 | 30 |
| | Upgrading Existing Applications and Scripts | 31 |
| | Upgrading Tool Scripts | 31 |
| | Upgrading Objectivity/C++ Applications | 31 |
| | Upgrading Objectivity/DB Active Schema for C++ Applications | 31 |
| | Upgrading Objectivity for Java Applications | 31 |
| | Upgrading Objectivity/DB Active Schema for Java Applications | 32 |
| | Maintaining Earlier Objectivity/DB Applications | 33 |
| | Maintaining Unrebuilt Release 10.0 or Release 10.1 Applications | 33 |
| | Maintaining Unrebuilt Release 9.x Applications | 33 |
| | Maintaining Unrebuilt pre-Release 9.0 Applications | 34 |

Getting Help

We have done our best to make sure all the information you need to install and operate each product is provided in the product documentation. However, we also realize problems requiring special attention sometimes occur.

How to Reach Objectivity Customer Support

You can contact Objectivity Customer Support by:

- **Telephone:** Call 1.408.992.7100 *or* 1.800.SOS.OBJY (1.800.767.6259) Monday through Friday between 6:00 A.M. and 6:00 P.M. Pacific Time, and ask for Customer Support.

The toll-free 800 number can be dialed *only* within the 48 contiguous states of the United States and Canada.

- **FAX:** Send a fax to Objectivity at 1.408.992.7171.
- **Electronic Mail:** Send electronic mail to help@objectivity.com.

Before You Call

Please be ready to submit the following to Objectivity Customer Support:

- Your name, company name, address, telephone number, fax number, and email address
- Detailed description of the problem you have encountered
- Information about your workstation environment, including the type of workstation, its operating system version, and compiler or interpreter
- Information about your Objectivity products, including the version of the Objectivity/DB libraries

You can use the Objectivity/DB `oosupportinfo` tool to obtain information about your workstation environment and your Objectivity products.

Release Overview

This release note describes the additions and changes made to Objectivity products and documentation in Release 10.1.2. This chapter provides an overview of these changes. This chapter summarizes:

- [Updated products](#)
- [New architectures](#)
- [New and updated books](#)

For Additional Information About This Release

Consult the *detailed release notes* (a series of plain text files), for the latest information about supported platforms and compilers, required operating-system patches and compiler patches, the open and fixed software problems, and documentation errata and corrections. You can find the detailed release notes for Release 10.1.2 on the Objectivity Technical Support Web site. Call Objectivity Customer Support to get access to this Web site.

For Information About Previous Releases

Release 10.1.2 includes the new features introduced in Release 10.0 and Release 10.1. For a summary of those features, see the *Objectivity Release Notes* (PDF) for these releases on the Technical Support Web site. Call Objectivity Customer Support to get access to this Web site.

New Products and Features

The primary purpose of this release is to fix defects, which are listed in the detailed release notes on the Objectivity Technical Support Web site. Call Objectivity Customer Support to get access to this Web site.

Updated Products

Products With New, Changed, or Removed Features

| Product Name | Product Description | See |
|---------------------------------------|--|-------------------------|
| Objectivity/DB | Distributed object database. | page 13 |
| Objectivity/C++ | C++ programming interface to Objectivity/DB. | page 16 |
| Objectivity for Java | Java programming interface to Objectivity/DB. | page 16 |
| Objectivity/DB Active Schema for Java | Java programming interface for reading and modifying an Objectivity/DB schema dynamically. | page 17 |
| Objectivity/.NET for C# | C# programming interface to Objectivity/DB. | page 18 |
| Objectivity/Python | Python programming interface to Objectivity/DB. | page 19 |

Products Rebuilt for Compatibility

| Product Name | Product Description |
|--|--|
| Objectivity/DB High Availability (Objectivity/HA) | Objectivity/DB option supporting autonomous partitions and data replication. |
| Objectivity/DB In-Process Lock Server (Objectivity/IPLS) | Objectivity/DB option that enables a C++, Java, or Smalltalk database application to run a lock server within the same process to improve performance. |
| Objectivity/DB Open File System (Objectivity/OFS) | Customizable interface between Objectivity/DB and hierarchic storage systems. |
| Objectivity/DB Parallel Query Engine (Objectivity/PQE) | Objectivity/DB option enabling applications to find persistent objects by searching separate portions of a federation in parallel. |
| Objectivity/DB Secure Framework (Objectivity/Secure) | Customizable interface between Objectivity/DB and standard security solutions. |

| Product Name | Product Description |
|--|--|
| Objectivity/DB Active Schema for C++ | C++ programming interface for reading and modifying an Objectivity/DB schema dynamically. |
| Objectivity/C++ Data Definition Language (Objectivity/DDL) | Objectivity/C++ option for creating and maintaining a schema of persistence-capable class definitions. |
| Objectivity/Smalltalk for VisualWorks | Smalltalk programming interface to Objectivity/DB. |
| Objectivity/SQL++ | Server and tools providing ANSI-standard SQL access to Objectivity/DB with object-oriented extensions to SQL. |
| Objectivity/SQL++ ODBC Driver (Objectivity/ODBC) | Objectivity/SQL++ option that enables ODBC-compliant client applications to access an Objectivity/DB federated database. |

New Platforms

No new architectures are introduced in this release.

NOTE New architectures were introduced in Release 10.0 and Release 10.1. See the *Objectivity Release Notes* (PDF) for those releases on the Technical Support Web site.

New and Updated Books

During installation, the online books for Objectivity products are placed in the doc subdirectory of your Release 10.1.2 Objectivity/DB installation directory.

NOTE All Objectivity online books are available on the Objectivity Technical Support Web site. Contact Objectivity Customer Support to get access to this Web site.

The following sections list the books delivered with this release.

Books in PDF

This section lists Objectivity books in Portable Document Format (PDF).

- New or updated for Release 10.1.2:
 - *Objectivity Release Notes, Release 10.1.2* (this document)
 - *Objectivity/DB Schema Development, Release 10.1.2*
- Earlier PDF books used with Release 10.1.2:
 - *Installation and Platform Notes for Windows, Release 10.1*
 - *Installation and Platform Notes for UNIX, Release 10.1*
 - *Installation and Platform Notes for Macintosh, Release 10.1*
 - *Objectivity/DB Administration, Release 10.1*
 - *Objectivity/DB High Availability, Release 10.1*
 - *Objectivity/C++ Programmer's Guide, Release 10.1*
 - *Objectivity/C++ Programmer's Reference, Release 10.1*
 - *Objectivity/C++ Data Definition Language, Release 10.1*
 - *Objectivity/DB Active Schema for C++, Release 10.1*
 - *Objectivity for Java Programmer's Guide, Release 10.1*
 - *Monitoring Lock Server Performance, Release 10.0*
 - *Objectivity/SQL++, Release 10.0*
 - *Objectivity/SQL++ ODBC Driver User's Guide, Release 10.0*
 - *Objectivity/Smalltalk for VisualWorks, Release 8*

Accessing PDF Books

After you install Objectivity/DB, you can access Objectivity books by clicking links from the following PDF file:

`installDir\doc\ObjyBooks.pdf` on Windows
`installDir/arch/doc/ObjyBooks.pdf` on UNIX and Macintosh

where:

`installDir` Release 10.1.2 Objectivity/DB installation directory.
`arch` UNIX or Macintosh architecture name.

Books in HTML

The following books are available in HTML format:

- New or updated for Release 10.1.2:
 - *Getting Started With Objectivity/Python, Release 10.1.2*
 - *Objectivity/Python Programmer's Reference, Release 10.1.2*
- Earlier HTML books used with Release 10.1.2:
 - *Objectivity/DB Administration, Release 10.1*
 - *Objectivity/DB High Availability, Release 10.1*
 - *Objectivity for Java Programmer's Guide, Release 10.1*
 - *Objectivity for Java Programmer's Reference, Release 10.1*
 - *Objectivity/DB Active Schema for Java Programmer's Reference, Release 10.1*

You can use your Web browser to access these books from the following HTML index files:

`installDir\doc\java\index.html` on Windows
`installDir\doc\objyHelp\index.html`
`installDir\doc\python\index.html`
`installDir/arch/doc/java/index.html` on UNIX and Macintosh
`installDir/arch/doc/objyHelp/index.html`
`installDir/arch/doc/python/index.html`

where:

`installDir` Release 10.1.2 Objectivity/DB installation directory.
`arch` UNIX or Macintosh architecture name.

Books in Compiled HTML Help

The *Objectivity/.NET for C# Programmer's Reference, Release 10.1* is provided as a Microsoft Compiled HTML Help file on the Windows platform.

- To open the help file, double-click on it:
`installDir\doc\objyNETcsharp.chm`

Advanced Topics

The **Documents** link on the Objectivity Technical Support Web site provides information about the following advanced topics.

| Document Title | Describes |
|-------------------------------------|--|
| Page and Object Encryption | How to provide plugin code to encrypt and decrypt persistent objects and pages. |
| External Query Agents | How to provide plugin code to extend the query server so that Objectivity predicate scans can fetch information from sources other than Objectivity databases. |
| Expression Tree Interface | The predicate expression tree interface, which provides support for external search agents and query builders and parsers. |
| Custom Operator Support | How to provide custom operators that can be used by the Objectivity predicate query language. |
| Parallel Query Engine Customization | How to set up and use custom parallel queries. |

Check the Technical Support Web site for additional advanced topics as they become available.

New and Changed Features

This chapter describes new and changed features of Objectivity products in Release 10.1.2.

NOTE Release 10.1.2 includes the new and changed features introduced in Release 10.0 and Release 10.1. For a summary of those features, see the *Objectivity Release Notes (PDF)* for those releases on the Technical Support Web site. Call Objectivity Customer Support to get access to this Web site.

Objectivity/DB (Database Services)

This section describes new, changed, and deprecated features of Objectivity/DB. See the Technical Support Web site for software or documentation problems that have been fixed in this release.

New and Changed Features

Enhanced Optimized Predicate Scans

In previous releases, an index defined on a particular storage object could be used to optimize predicate scans with multiple optimized conditions only if the conditions tested the first n key fields of the index *in the correct order* (where n is an integer greater than one and less than or equal to the number of key fields).

In Release 10.1.2, the order of the optimized conditions in the predicate scan need not match the order of the index keys to obtain the optimization.

New Name Map Operator

The predicate query language has a new *name map* operator that produces the value corresponding to the specified key of a name map, which is either a reference attribute or a to-one association attribute.

| Operator | Description | Usage | First Operand (op1) | Second Operand (op2) | Result Type |
|----------|--|------------------------|------------------------------------|----------------------|-------------|
| [' '] | Returns the value corresponding to the <code>op2</code> key in the name map. | <code>op1 [op2]</code> | Reference to an <code>ooMap</code> | String | Reference |

A name map operator cannot be the sole operator in a predicate string, but must be combined and nested with other operators. For example, the following expression evaluates to `true` if the `vehiclesList` attribute references a name map that includes a valid value for the license key string, `AR698L`.

```
isValid(vehiclesList['AR698L'])
```

Bypassing Objectivity/DB Name Resolution on UNIX Platforms

The `oo_local_host` keyword, which enables you to bypass Objectivity/DB name resolution, is now exposed for general use on UNIX platforms.

You can specify the `oo_local_host` keyword as the *host* value to applications and tools running on UNIX client hosts, when designating Objectivity/DB files in certain network configurations. This keyword causes the corresponding *path* value to be interpreted entirely by the client host's operating system, instead of allowing Objectivity/DB to process the *path* value first to extract host information from it using the local mount table.

Specifying the `oo_local_host` keyword in a UNIX configuration may be appropriate in scenarios such as the following:

- You are running an application or tool that needs to access files on a special-purpose storage device that is not running an NFS server that Objectivity/DB can work with directly.
- You are running an application or tool on one or more processors in a cluster, where these processors are to be considered interchangeable, and you do not want Objectivity/DB catalogs to record any particular processor's name as the host name of a file being created on the cluster's local disk.

Objectivity/DB (Tools)

This section describes new, changed, deprecated, and obsolete features of Objectivity/DB tools. See the Technical Support Web site for software or documentation problems that have been fixed in this release.

New and Changed Features

Tools with new or changed features are listed in alphabetical order by tool name.

oocheck and oofix Enhancements

The `oocheck` and `oofix` tools can now be used independently of any particular release of Objectivity/DB.

- These tools can now be used with any federated database, regardless of the release for which it is licensed.
- These tools are now statically built so you can run them whether or not the current version of Objectivity/DB is installed.

These enhancements make it easier for an administrator to check and fix an older, deployed federated database in an end-user environment.

Objectivity/C++

This section describes new, changed, deprecated, and obsolete features of Objectivity/C++. See the Technical Support Web site for software or documentation problems that have been fixed in this release.

New and Changed Features

Iterator for To-Many Associations Accepts an Object Qualifier

A new `linkName` generated method lets you provide an object qualifier for qualifying destination objects of a to-many association.

```
ooStatus linkName(
    ooItr(className) &iterator,
    ooMode openMode,
    const ooObjectQualifier& objQualifier);
```

Objectivity for Java

This section describes new, changed, deprecated, and obsolete features of Objectivity for Java. See the Technical Support Web site for software or documentation problems that have been fixed in this release.

New and Changed Features

Relationship-Persistence Policy Has Pre-Release 10.0 Behavior

Transient source objects of unidirectional relationships are once again subject to the relationship-persistence policy set by the following method:

- `com.objy.db.app.Session.setFormTransientRelationships(boolean)`

This change reinstates the pre-Release 10.0 behavior, in which a transient source object of a unidirectional relationship remains transient only if the application has previously called the session's `setFormTransientRelationships` method with the parameter set to `true`; otherwise, the transient source object is automatically made persistent at the time the relationship is formed.

For two releases (Release 10.0 and Release 10.1), the relationship-persistence policy did *not* apply to unidirectional relationships with transient source objects if they also had persistent destination objects. During these releases, forming a unidirectional relationship from a transient source object to a persistent

destination object caused the source object to be made persistent immediately, regardless of the relationship-persistence policy in effect.

Reinstating the pre-Release 10.0 behavior affects your application if it:

1. Calls a session's `setFormTransientRelationships` method with the parameter `true`—for example, to enable transient objects to be added to a transient name map.
2. Forms a unidirectional relationship (in the same session) from a transient source object to a persistent destination object. The source object remains transient due to step 1.
3. Attempts an operation on the source object that can be performed only on a persistent object, such as giving it a scope name or adding it to a persistent scalable collection.

If your application relies on the Release 10.0/Release 10.1 behavior to automatically persist the relevant transient objects irrespective of the relationship-persistence policy, you can obtain that behavior by calling the following method of the session:

```
session.  
    setPersistUnreachableTransientsWithRelationshipToPersistent (  
        true);
```

This is considered a temporary workaround, however. The recommended fix is to add code to make the relevant transient source object persistent after step 2, before attempting the operation in step 3.

Objectivity/DB Active Schema for Java

This section describes new, changed, deprecated, and obsolete features of Objectivity/DB Active Schema (Active Schema) for Java. See the Technical Support Web site for software or documentation problems that have been fixed in this release.

Obsolete Features

VArray_Object and Relationship_Object Default Constructors

The following default constructors should no longer be used:

- `com.objy.as.app.VArray_Object()`
- `com.objy.as.app.Relationship_Object()`

A call to either of these constructors now throws an `ObjyRuntimeExcepTion`. An application that uses either of these default constructors should be changed to use the class's copy constructor instead.

Objectivity/.NET for C#

This section describes new, changed, deprecated, and obsolete features of Objectivity/.NET for C#. See the Technical Support Web site for software or documentation problems that have been fixed in this release.

New and Changed Features

New Debug Assembly

A debug version of the Objectivity/DB assembly `Objectivity.DBxxx.dll` is now provided in the following location of the Objectivity/DB installation directory *installDir*:

```
installDir\bin\debug
```

New Method for Obtaining an ObjectId

You can obtain an object identifier (an instance of the `ObjectId` struct) with the specified component values by calling the following new static method:

- `ObjectId.FromString`

You specify the desired component values as a string in the format "#*D-C-P-S*", where *D* is a database identifier, *C* is a container identifier, *P* is a logical page number, and *S* is a logical slot number.

Objectivity/Python

This section describes new, changed, deprecated, and obsolete features of Objectivity/Python. See the Technical Support Web site for software or documentation problems that have been fixed in this release.

New and Changed Features

Ability to Get Type Number From a Persistent Object

The following new method enables you to get the type number of a persistent object's class directly from that object:

- `Obj::typeName()`

Ability to Cluster Python Objects According to OID

Objectivity/Python methods now allow you to specify an OID as the placement argument, as an alternative to specifying either a persistent object or a persistent object with the `as oocOnNewPage` clustering rule. For example, the following creates and returns a new instance of `myUserClass` placed according to the specified OID.

```
oopython.addObject("2-3-554-7", "myUserClass")
```

The following methods accept an OID as the placement argument:

- `OpObjy::addObject`
- `OpObjy::addTreeListX`
- `OpObjy::addTreeMapX`
- `OpObjy::addTreeSetX`
- `OpObjy::addHashSetX`
- `OpObjy::addHashMapX`
- `OpObjy::addVector`
- `OpObjy::addKeyDesc`
- `OpObjy::addMap`

Adding Methods to Python Classes

Objectivity/Python lets you add methods to any application-defined, persistence-capable class. To do this, make your class inherit from the class `Persistent`, which is now exposed for general use, then create a Python initializer that calls the `makePersistent` inherited method.

Your class can now be instantiated using the initializer, and its methods can be called.

The following class description shows a user-defined, persistence-capable class that includes a `printTest` method.

```
>>> class Person(oopython.Persistent):
...     def __init__(self, container, name="", status=1):
...         self.persist(container)
...         self.name = name
...         self.status = status
...     def persist(self, container):
...         self.makePersistent(container)
...     def printTest(self):
...         print "\n\n*** Hello world!\n"
```

Release Compatibility

This chapter provides information about the impact, if any, of using Release 10.1.2 of Objectivity/DB with existing data, tools, or applications from an earlier release. You may need to perform an upgrade or be aware of limitations.

- See “Upgrading Existing Federated Databases” below if you plan to use tools or applications built with the current release to access data created with an earlier release.
- See “Upgrading Existing Applications and Scripts” on [page 31](#) if you plan to rebuild existing applications with the current release.
- See “Maintaining Earlier Objectivity/DB Applications” on [page 65](#) if you plan to continue using unrebuilt tools or applications from an earlier release.

Upgrading Existing Federated Databases

Federated databases created with Release 8.x or Release 9.x of Objectivity/DB can be upgraded to match the new release.

At a minimum, an existing federated database requires an Objectivity Release 10 license before you can access the data in it with any of the following:

- Tools provided with Objectivity/DB Release 10.1.2
- New applications built with Objectivity/DB Release 10.1.2
- Existing applications that have been upgraded, recompiled, and relinked with Objectivity/DB Release 10.1.2 (see “Upgrading Existing Applications and Scripts” on [page 31](#)).

Other upgrades are optional, depending on the features and products to be used. See “Selecting the Upgrade Tasks to Perform” on [page 22](#).

Selecting the Upgrade Tasks to Perform

Use the following tables to determine which upgrade tasks to perform on a federated database that has not previously been upgraded.

To upgrade a Release 10.x federated database

NOTE No upgrades are required.

To upgrade a Release 9.3 or 9.4 federated database

| Do This | If You Want To | See |
|--------------------------------|---|-------------------------|
| Update the Objectivity license | Access the federated database with Release 10.1.2 | page 23 |

To upgrade a Release 9.0, 9.1, or 9.2 federated database

| Do This | If You Want To | See |
|---|---|-------------------------|
| Update the Objectivity license | Access the federated database with Release 10.1.2 | page 23 |
| Upgrade the federated database's schema | Use features introduced in Release 9.3 | page 30 |

To upgrade a Release 8.x federated database

| Do This | If You Want To | See |
|--|---|-------------------------|
| Update the Objectivity license | Access the federated database with Release 10.1.2 | page 23 |
| Upgrade the federated database's internal database format and schema | Use features enabled by the new internal database format introduced in Release 9.0, and the schema introduced in Release 9.3 You must perform this upgrade if you want to use the backup & restore tools introduced in Release 10.1. | page 24 |

NOTE If you need to update the Objectivity license, you must do so before performing any other upgrades.

Updating to an Objectivity Release 10 License

A Release 8.x or Release 9.x federated database must have an Objectivity Release 10 license to authorize access by Release 10.1.2 tools and applications. If you have not already done so, you must perform the following steps to replace the federated database's existing license with your Objectivity Release 10 license.

NOTE An Objectivity Release 10 license authorizes access by all Release 10.x tools and applications.

To update the Objectivity license for one or more Release 8.x or Release 9.x federated databases:

1. Verify that you have set up a default license file containing your Objectivity Release 10 license. See the Objectivity/DB installation chapter of *Installation and Platform Notes* for your platform.
2. For each Release 8.x or Release 9.x federated database to be accessed with Release 10.1.2 tools and applications, enter:

```
oolicense -fromdefault bootFilePath
```

where

`-fromdefault` Obtains `oolicense.txt` in the Objectivity/DB installation directory `installDir`.

`bootFilePath` Path to the boot file of the federated database. You can omit this parameter if you set the `OO_FD_BOOT` environment variable to the correct path.

Upgrading to the Release 9.0 Internal Database Format

For backward compatibility, Release 10.1.2 tools and applications can access the data in a federated database created with Release 8.x, provided the license is updated. However, you must upgrade the internal database format of a Release 8.x federated database if you want the tools and applications accessing it to take advantage of:

- The new backup and restore tools introduced with Release 10.1.
- Various features introduced in Release 9.x and Release 10.0. Such features are described in *Objectivity Release Notes* for each release, available on the Technical Support Web site.

About the Upgrade Procedure

The upgrade procedure uses the Release 10.1.2 tool `ooupgrade` to convert the internal database format of an existing Release 8.x federated database. This procedure additionally updates the federated database's schema to support features introduced in Release 9.3.

NOTE You do not need to upgrade the internal database format of a Release 9.x federated database. If appropriate, you can upgrade the federated database's schema as described in "Upgrading the Schema to Release 9.3" on [page 30](#).

You can upgrade an entire federated database in a single operation or one database at a time.

Each database is upgraded "in place." The original database is renamed with a tilde appended to the original filename. A database with the new format is created using the original filename, and is then populated with the information from the original database. The renamed original database is deleted, leaving the now-upgraded database with the original filename. The upgrade operation preserves the identifiers of all objects in an upgraded database.

WARNING Once a federated database is upgraded, it cannot be accessed by pre-Release 9.0 tools and applications. If interoperating with such tools and applications is required, skip this upgrade procedure and see "Maintaining Earlier Objectivity/DB Applications" on [page 33](#).

WARNING The upgrade operation deletes any backup set entries and information about previous backups. The first backup after an upgrade should be a full backup.

Preparing a Federated Database for Upgrade

Before upgrading all or part of a Release 8.x federated database, prepare it as follows:

1. If you have not already done so:
 - a. make sure Release 10.1.2 is completely installed. Update the Objectivity license in the existing federated database; see [page 23](#).
 - b. Make a full backup. (Use the `oobackup` tool, *not* the new `oobackupx` tool.) If you plan to upgrade one database at a time, the safest procedure is to make a full backup before you upgrade each database.
2. Run the `oocheck` tool to verify the federated database is consistent. You will need to fix any inconsistencies before upgrading. For example, you could use the `oofix` tool or call Objectivity Customer Support.
3. If the existing federated database is partitioned, read “Clearing Autonomous Partitions Before Upgrading” on [page 29](#), and clear autonomous partitions as necessary.
4. Ensure that the lock-server host for the federated database being upgraded is running a lock server from Objectivity/DB Release 7.1 or later. This is required for compatibility with the new internal database format.

Preparing to Run `ooupgrade`

To prepare to run `ooupgrade` on a Release 8.x federated database:

1. If you have not already done so, make a full backup.
2. Be sure you have enough available disk space for the upgrade.

Upgrading a database requires temporary access to free space equal to the database’s original size. If multiple databases are being upgraded, the available free space must equal the size of the largest database. You need more free space if you choose an architecture in step 3 that causes conversion from a 32-bit disk format to a 64-bit disk format, or if you increase the storage-page size in step 4.
3. (Optional) Choose a different owning architecture for the storage pages of each database being upgraded. The architecture of a storage page determines the disk format used by the page.

By default, the storage pages of an upgraded database retain their original owning architecture(s), except for the system database, which is converted to the disk format of the current host architecture. **Hint:** You can run `ooupgrade -help` to get a list of available architectures.

4. (Optional) Choose a larger storage-page size for each database being upgraded. For guidelines, see “Increasing the Storage-Page Size” on [page 26](#).
5. Arrange to stop applications from accessing the federated database while the system database is being upgraded. After the system database is upgraded, concurrent access can be resumed in databases other than the database currently being upgraded. (The `ooupgrade` tool obtains an exclusive lock on the database currently being upgraded.)

Increasing the Storage-Page Size

By default, an upgrade operation preserves the original storage-page size used by each upgraded database. You can choose to increase the storage-page size of an upgraded database for reasons such as the following:

- If the database contains storage pages that are owned by a 32-bit architecture, and the upgrade operation will change the ownership to a 64-bit architecture.
- If the database contains storage pages that are currently full, and you want applications to be able to create additional persistent objects on those storage pages after the upgrade operation is complete.

NOTE Increasing the size of the storage pages in a database causes the overall size of the database file to increase. You may not decrease the storage-page size.

If you increase the storage-page size while upgrading multiple databases in a single operation, all of the upgraded databases will use the same storage-page size. If only a subset of your databases require a larger storage-page size, consider upgrading them separately, before upgrading the remaining databases as a group, preserving their original storage-page sizes.

If the new storage-page size is greater than the default storage-page size of the federated database, then the default storage-page size of the federated database (and the system database) is changed to the new value.

Increasing a database’s storage-page size is recommended if the database contains storage pages owned by a 32-bit architecture, and you have chosen a 64-bit architecture in step 3 above. Such a change increases the sizes of basic objects on the converted storage pages. The upgrade operation automatically *redirects* objects to pages with more storage space as needed to accommodate the resized basic objects. Redirection can have a minor performance impact for operations that access the objects.

You can avoid such redirection by specifying a larger storage-page size for the upgrade. The amount by which to increase a database’s storage-page size depends on the data itself. Each reformatted basic object will require an additional 8 bytes of overhead, plus 4 bytes for each inline to-many association (if

any), plus 4 bytes for each variable-sized array (VArray) attribute (if any). One conservative approach for avoiding redirection is to use one and one half times the current page size, if this is possible without exceeding the maximum possible page size.

Increasing a database's storage-page size enables `oouprgrade` to eliminate existing redirection for:

- Basic objects that were previously redirected as a result of schema evolution. Each such object is moved back to the page specified in its object identifier, provided that page has sufficient space after being resized.
- Variable-size arrays (VArrays) that were previously redirected as a result of acquiring too many elements to fit on their original pages. Each such VArray is moved back to the same page as the object that owns it, provided the page has sufficient space after being resized.

Running the `oouprgrade` Tool

WARNING

Proceed only if you have a valid backup of the federated database or database being upgraded.

Enter `oouprgrade` with the indicated options:

- To upgrade an entire federated database in a single operation, enter:

```
oouprgrade -all bootFilePath
```

where

`-all` Upgrades the entire federated database, including the system database and every database.

`bootFilePath` Path to the boot file of the federated database.

- To upgrade a federated database one database at a time, enter the following for each database:

```
oouprgrade -db dbSysName bootFilePath
```

where

`-db dbSysName` Upgrades an individual database specified by the system name. If the system database has not yet been upgraded, it is automatically upgraded with the first database you specify.

`bootFilePath` Path to the boot file of the federated database.

In either case, you may also need to include one or more of the following options:

| | |
|------------------------------------|--|
| <code>-licensefile fileName</code> | File containing your current Objectivity license. The <code>ooupgrade</code> tool must be able to access this file to refresh the license in the upgraded system database. You can omit this option if your license file is in a default location. |
| <code>-format newArch</code> | New owning architecture for the storage pages in each database being upgraded (including the system database), if a different owning architecture is desired. |
| <code>-pagesize size</code> | Storage-page size, in bytes, for each database being upgraded, if a larger page size is required. See “Increasing the Storage-Page Size” on page 26 |

After the Upgrade Operation Completes

After a federation database has been upgraded:

- No pre-Release 9.0 application or tool can access the upgraded federated database.
- The upgrade operation deletes any backup set entries and information about previous backups. Consequently, the first backup after an upgrade should be a full backup.

Note: You can use either the superseded `oobackup` tool or the Release 10.1 `oobackupx` tool.
- The federation database’s schema is up-to-date, and contains descriptions of all the system schema types that were added in Release 9.0 and Release 9.3.

Note: The upgrade operation has no effect on the schema descriptions of application-specific classes. Consequently, if an application-specific class was added to the schema in Release 8.x, any persistent character or Boolean attributes in the class will continue to be represented using Objectivity/DB integer types.
- Any new databases subsequently added to the federated database will use the new internal database format.

Optional Processing After Increasing the Storage-Page Size

If you performed an upgrade operation that increased the storage-page size in one or more databases of a federated database, you can optionally write and run a post-processing utility to ensure that every object takes advantage of the increased storage-page size. Doing so can improve the performance of

subsequent application access to the federated database. Your utility can perform any of the following tasks as appropriate:

- Re-create existing persistent collections.
The storage-page size of a database sets the maximum size of node-arrays in an ordered collection, and is a factor for determining the maximum capacity of hash buckets in an unordered collection. Increasing the storage-page size with `ooupgrade` does not affect these limits for existing persistent collections. If you want the node-arrays (or hash buckets) of an existing persistent collection to take advantage of the increased storage-page size, your utility must create a new, replacement persistent collection, and call the replacement collection's `addAll` method, passing the original collection as the parameter. Make sure the utility uses an appropriate clustering strategy for placing the nodes (or hash buckets) of the new collection.
- Drop and re-create indexes.
The storage-page size of a database sets the maximum size of an index's internal data structure. Increasing the storage-page size with `ooupgrade` does not affect this limit for existing indexes. If you want an existing index to take advantage of the increased storage-page size, your utility must obtain the appropriate key description, and use it to drop and re-create the index.

Clearing Autonomous Partitions Before Upgrading

Earlier releases of Objectivity/DB High Availability allowed you to transfer the control of a container from its home autonomous partition (the partition controlling the container's database) to a different partition. This feature is not supported in Release 9.0 (or later) federated databases.

Before you upgrade a partitioned federated database in which control of any container has been transferred, you must perform the following step:

- Run the `ooCLEARAP` tool on every partition that controls a transferred container.

Clearing an autonomous partition physically returns each container controlled by the partition to its containing database. For more information about `ooCLEARAP`, see *Objectivity/DB High Availability*.

Upgrading the Schema to Release 9.3

For backward compatibility, Release 10.1.2 tools and applications can access the data in a federated database created with earlier releases provided the license is updated. However, new tools and applications accessing a pre-Release 9.3 federated database cannot take advantage of certain features introduced in Release 9.3. In particular, Release 9.3 introduced enhanced scalable persistent collections. (For information about these collections, see *Objectivity Release Notes, Release 9.3*, on the Technical Support Web site.)

To take advantage of the features introduced in Release 9.3, you can upgrade the federated database's schema as described below. You normally perform this procedure only on a Release 9.0, Release 9.1, or Release 9.2 federated database.

NOTE You do not need to explicitly upgrade the schema of a Release 8.x federated database if you are upgrading it as described in "Upgrading Existing Federated Databases" on [page 21](#).

To upgrade the schema of a pre-Release 9.3 federated database:

1. Make a backup of the federated database to be upgraded.
2. At a command prompt, enter:
 - On Windows:


```
ooschemaupgrade installDir\etc\schema.9.3.dmp
                bootFilePath
```
 - On UNIX:


```
ooschemaupgrade installDir/arch/etc/schema.9.3.dmp
                bootFilePath
```

where

installDir Your Release 10.1.2 Objectivity/DB installation directory.

arch Architecture name for your platform (UNIX only).

bootFilePath Path to the boot file of the federated database to be upgraded.

Upgrading Existing Collections

Release 10.1.2 applications and tools can access an existing federated database that contains pre-Release 9.3 persistent collections, and can create new enhanced scalable persistent collections, provided you have upgraded the federated database's schema. Although both new and old persistent collections can coexist in the same federated database, they are accessible only by separate portions of your application code.

You can upgrade an existing pre-Release 9.3 persistent collection by writing a special-purpose application that creates a new, enhanced persistent collection as a replacement for the existing one. For guidelines for writing such an application, see *Objectivity Release Notes, Release 9.3*, on the Technical Support Web site.

Upgrading Existing Applications and Scripts

You can upgrade an existing application to take advantage of this release's features and fixes. To upgrade an application, you must recompile it and relink it with Release 10.1.2 libraries.

When planning whether to upgrade existing applications to Release 10.1.2, you should take into account any required code changes listed in the following subsections.

NOTE These subsections describe only the changes introduced in Release 10.1.2. For descriptions of code and script changes introduced in earlier releases, see the *Objectivity Release Notes* for Release 10.1, Release 10.0, and Release 9.x on the Technical Support Web site.

Upgrading Tool Scripts

Release 10.1.2 does not introduce any changes that require you to rewrite portions of existing tool scripts.

Upgrading Objectivity/C++ Applications

Release 10.1.2 does not introduce any changes that require you to rewrite portions of existing Objectivity/C++ applications.

Upgrading Objectivity/DB Active Schema for C++ Applications

Release 10.1.2 does not introduce any changes that require you to rewrite portions of existing Objectivity/DB Active Schema for C++ applications.

Upgrading Objectivity for Java Applications

Use the following list to determine whether you must rewrite portions of existing Java applications to accommodate Release 10.1.2 changes in Objectivity for Java.

- Read "Relationship-Persistence Policy Has Pre-Release 10.0 Behavior" on [page 16](#) to determine whether your application will be affected by this

change. If it is, you can use the temporary workaround or performed the recommended fix described in that section.

Upgrading Objectivity/DB Active Schema for Java Applications

Use the following list to determine whether you must rewrite portions of existing Java applications to accommodate Release 10.1.2 changes in Active Schema for Java.

- Replace any code for the obsolete item listed on [page 17](#). In particular, any call to the default constructor of the `com.objy.as.app.VArray_Object` or `com.objy.as.app.Relationship_Object` class must be replaced by a call to the class's copy constructor.

Maintaining Earlier Objectivity/DB Applications

After installing Objectivity/DB Release 10.1.2 along with your chosen Objectivity programming interface, you normally:

- Develop new Release 10.1.2 applications.
- Upgrade existing applications and then recompile and relink them with Release 10.1.2; see “Upgrading Existing Applications and Scripts” on [page 61](#).

In some situations, you may also need to maintain existing applications built with an earlier (pre-Release 10.1.2) Objectivity/DB release.

Maintaining Unrebuilt Release 10.0 or Release 10.1 Applications

If you are maintaining an unrebuilt Release 10.0 or Release 10.1 application, note that such earlier applications:

- Are completely compatible with Release 10.1.2 federated databases, with no limitations.
- Can interoperate with new or upgraded applications, with the no limitations:

Maintaining Unrebuilt Release 9.x Applications

If you are maintaining an unrebuilt Release 9.x application (that is, an application built with Objectivity/DB Release 9.0, Release 9.1, Release 9.2, Release 9.3, or Release 9.4), note that such earlier applications:

- Are completely compatible with Release 10.1.2 federated databases, with the following limitation:
 - An unrebuilt earlier application cannot access a Release 10.1.2 federated database or autonomous partition with a boot-file name that has only 1 character, or has 32 or more characters.
- Can interoperate with new or upgraded applications, with the following limitation:
 - An unrebuilt pre-Release 9.3 application cannot access any enhanced collections created by a new or upgraded application.

Maintaining Unrebuilt pre-Release 9.0 Applications

If you are maintaining an unrebuilt pre-Release 9.0 application, you should consult the section on maintaining earlier Objectivity/DB applications in *Objectivity Release Notes, Release 9.0*, on the Technical Support Web site.

An unrebuilt pre-Release 9.0 application:

- Is not compatible with Release 10.1.2 federated databases.
- Cannot interoperate with Release 10.1.2 applications unless the newer applications follow certain guidelines.
- Does not recognize the client-host architectures introduced in Release 9.x or in Release 10.x.



OBJECTIVITY, INC.

640 West California Avenue, Suite 210

Sunnyvale, California 94086-3624

USA

+1 408.992.7100

+1 408.992.7171 Fax

www.objectivity.com

info@objectivity.com