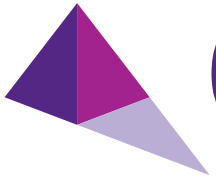

Objectivity/DB

Objectivity Release Notes

Release 11.2

Objectivity Release Notes

Part Number: 11.2-RN-0

Release 11.2, August 29, 2013

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 1993–2013 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., 3099 North First Street, Suite 200, San Jose, CA 95134.

Contents

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.

1

Release Overview

This release note describes the additions and changes made to Objectivity products and documentation in Release 11.2. This chapter provides an overview of these changes. This chapter summarizes:

- New products and features
- Deprecated feature
- Updated products
- New and updated books

For Additional Information About This Release

The Objectivity Developer Network has the latest detailed information about supported platforms and compilers, required operating-system patches and compiler patches, the open and fixed software problems, and documentation errata and corrections. Call Objectivity Customer Support to obtain access to this information, if you do not already have an account.

For Information About Previous Releases

Release 11.2 includes the new features introduced in Release 11.0. For a summary of these features, see the corresponding *Objectivity Release Notes* on the Objectivity Developer Network.

New Products and Features

The following table lists the new Objectivity products and features in this release.

Product or Feature	Description	See
General predicate query language (PQL)	New operators and enhanced, reorganized PQL documentation.	page 14
Tool-runner based tools	New tools available through the tool-runner command-line mechanism, which accepts Unicode UTF-8 string values.	page 17
Programmatic tool execution	(*Objectivity/C++*) Programming interface for invoking tool-runner based tools from an application.	page 20
Enhanced navigation query	(*Objectivity/C++*) Enhancements to path processing in navigation queries.	page 20
Predicate query language (PQL) for navigation queries	(*Objectivity/C++*) New navigation-path operators, and updates to existing operators to support navigation queries.	page 21

Deprecated Feature

The following table lists Objectivity features that are deprecated in this release.

Feature	Description	See
Pre-Release 9.0 internal database format	Internal format used for catalogs of databases and containers, in federated databases created before Objectivity/DB Release 9.0.	page 19

NOTE Starting with Release 11.3, new or rebuilt applications will no longer be able to access a pre-Release 9.0 federated database unless it is upgraded to use the Release 9.0 (or later) internal database format.

Updated Products

Products With New, Changed, or Removed Features

Product Name	Product Description	See
Objectivity/DB	Distributed object database.	page 14, page 17
Objectivity/C++	C++ programming interface to Objectivity/DB.	page 20
Objectivity for Java	Java programming interface to Objectivity/DB.	page 23

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 (Lock-Server Performance Monitor)	Programming interface for writing C++ programs that gather information about how database applications use a running Objectivity/DB lock server.
Objectivity/DB Open File System (Objectivity/OFS)	Customizable interface between Objectivity/DB and hierarchic storage systems.
Objectivity/DB Secure Framework (Objectivity/Secure)	Customizable interface between Objectivity/DB and standard security solutions.
Objectivity/C++ Data Definition Language (Objectivity/DDL)	Objectivity/C++ option for creating and maintaining a schema of persistence-capable class definitions.
Objectivity/.NET for C#	C# programming interface to Objectivity/DB.
Objectivity/Python	Python 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 and Updated Books

During installation, the online books for Objectivity products are placed in the following subdirectory of your Release 11.2 Objectivity/DB installation directory *installDir*:

installDir/doc

NOTE Objectivity books are also available on the Objectivity Developer Network.

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 11.2:
  - *Objectivity Release Notes, Release 11.2* (this document)
  - *Objectivity/DB Predicate Query Language*, *Release 11.2* — new book
  - *Objectivity/DB Administration*, *Release 11.2*
  - *Getting Started With Managed Object Placement*, *Release 11.2*
  - *Objectivity/C++ Programmer's Guide*, *Release 11.2*
  - *Objectivity/C++ Programmer's Reference*, *Release 11.2*
  - *Objectivity for Java Programmer's Guide, Release 11.2*
- Earlier PDF books used with Release 11.2:
  - *Objectivity/DB High Availability, Release 11.0*
  - *Monitoring Lock Server Performance, Release 11.0*
  - *Objectivity/C++ Data Definition Language*, *Release 11.0*
  - *Objectivity/DB Active Schema for C++*, *Release 11.0*
  - *Objectivity/DB Schema Development, Release 11.0*
  - *Objectivity/SQL++*, *Release 11.0*

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

where:

installDir Release 11.2 Objectivity/DB installation directory.

Books in HTML

The installation and configuration documentation is available on the Objectivity Developer Network.

The following books are available in HTML format in the Objectivity/DB installation directory:

- *Objectivity for Java Programmer's Guide,* Release 11.2
- *Objectivity for Java Programmer's Reference,* Release 11.2
- *Objectivity/DB Active Schema for Java Programmer's Reference,* Release 11.0
- *Getting Started With Objectivity/Python,* Release 11.0
- *Objectivity/Python Programmer's Reference,* Release 11.0

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

installDir/doc/java/index.html
installDir/doc/python/index.html

where:

installDir Release 11.2 Objectivity/DB installation directory.

Books in Compiled HTML Help

The *Objectivity/.NET for C# Programmer's Reference, Release 11.0* 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

Additional Topics

See the Objectivity Developer Network for the following:

- The PDF book *Objectivity/C++ Backward Compatibility,* which describes selected superseded Objectivity/C++ programming mechanisms.
- The advanced topics listed below:

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.
Server Administration Security	How to provide plugin code that implements security restrictions on selected server operations, such as stopping the lock server.

Other advanced topics are added as they become available.

2

New and Changed Features

This chapter describes new and changed features of Objectivity products in Release 11.2.

NOTE Release 11.2 includes the new and changed features introduced in Release 11.0. For a summary of these features, see the corresponding *Objectivity Release Notes* on the Objectivity Developer Network.

See the Objectivity Developer Network for software or documentation problems that have been fixed in this release. Call Objectivity Customer Support to obtain access to this information, if you do not already have an account.

Objectivity/DB (Database Services)

This section describes new, changed, and deprecated features of Objectivity/DB.

New and Changed Features

General Predicate Query Language (PQL) Enhancements

General updates to the Objectivity/DB predicate query language (PQL) include:

- New PQL operators.
- Reorganized and enhanced documentation, now in a single book.

NOTE Additional PQL operators are available for use in Objectivity/C++ navigation queries; see "Navigation-Path Operators" on page 21.

New PQL Operators

The following new PQL operators are available.

Operators	Description	Usage
ABS	New operator that produces the absolute value of a number.	`ABS(`*`op1`*`)` where *op1* is a numeric operand.

QUALIFY	New operator that tests whether a given object matches the specified type and predicate, or determines whether the current object being qualified matches the given type and predicate.	`QUALIFY(op1, op2, op3)`
		`QUALIFY(op2, op3)`
		where:
		op1 is a reference
		op2 is a class type
		op3 is a boolean.

The `ABS` math operator is useful when performing equality comparisons with floating-point numbers—for example:

```
"ABS(myFloatNumber - 78.01) < 0.001"
```

The `QUALIFY` operator can be used in object qualification as well as in Objectivity/C++ navigation-path qualification.

New Date-and-Time PQL Operators

The following new date-and-time operators are available.

Operator	Description	Usage	Unary Operand (op1)	Result Type
DAY_NAME	Returns the name of the day in all uppercase letters in English	DAY_NAME(*op1*)	Datetime, Date	String
MONTH_NAME	Returns the name of the month in all uppercase letters in English	MONTH_NAME(*op1*)	Datetime, Date	String
DAY_OF_WEEK	Returns the number of the day within the week	DAY_OF_WEEK(*op1*)	Datetime, Date	Integer
DAY_OF_MONTH	Returns the number of the day within the month	DAY_OF_MONTH(*op1*)	Datetime, Date	Integer
WEEK	Returns the number of the week within the year	WEEK(*op1*)	Datetime, Date	Integer
MONTH	Returns the number of the month within the year	MONTH(*op1*)	Datetime, Date	Integer
YEAR	Returns the year	YEAR(*op1*)	Datetime, Date	Integer

The DAY_NAME and MONTH_NAME operators accept a date or a date time and return the full name of the day or the month, respectively, in all uppercase letters in English.

The DAY_OF_WEEK operator returns the number of the day within the week, ranging from zero to six where Sunday is zero, Monday is one, and so forth.

The DAY_OF_MONTH operator returns the number of the day within the month.

The WEEK, MONTH, and YEAR operators return integers representing the number of a week within a year (based on the ISO 8601 standard), the number of a month within a year, or the year itself, respectively.

Enhanced PQL Type-Evaluation Operators

A new format for the KIND_OF operators is available. This format can accept a single class-type operand and return a Boolean value based on whether or not the current object is a kind of the given class type, where the current object is an

element in any multi-element, including an array of objects returned by an Objectivity/C++ navigation-path operator.

Operators	Description	Usage

`KIND_OF`
`IS_TYPE`
`IS`
	New functional format for the type-evaluation operators that returns true if the current object being qualified matches the provided class type.	`KIND_OF(`*`op1`*`)`
		`IS_TYPE(`*`op1`*`)`
		`IS(`*`op1`*`)`
		where:
		`op1` is a class type

The following example returns true if any vertex in a navigation result path has the type `Email`.

```
ANY(VERTICES(), IS_TYPE(CLASS:Email))
```

Enhanced Index Subscript Operator

The *index subscript operator* can now accept a negative number for indexing backwards from the last element in the index.

Operator	Description	Usage	First Operand (op1)	Second Operand (op2)[1]	Result Type[2]
`[]`	Returns an element at an index location	`op1[op2]`	Multi-element	Integer	Element type of op1

1. The index subscript operator can accept a negative number.
2. The index subscript operator returns null if either operand is a null value., or if the specified index is out of bounds.

When supplying an integer value to the index subscript operator:

- Zero accesses the first element in the index.
- One accesses the second element in the index, two accesses the third element, and so on.
- Negative one accesses the last element in the index, negative two accesses the element prior to that, and so on.

New PQL Documentation

PQL information is now consolidated into a single book called *Objectivity/DB Predicate Query Language*, which applies to all Objectivity/DB programming languages. The new book provides usage information for object qualification and navigation-path qualification, as well as complete reference information for PQL.

You can find the new book in the following subdirectory of your Objectivity/DB installation directory:

installDir/doc/queryLanguage.pdf

NOTE Usage and reference information for PQL has been removed from both *Objectivity/C++ Programmer's Guide* and *Objectivity for Java Programmer's Guide*.

Objectivity/DB (Tools)

This section describes new, changed, deprecated, and obsolete features of Objectivity/DB tools.

New and Changed Features

New Tools Available in Tool Runner

New versions of selected Objectivity/DB administration tools are now available in the tool runner. Tools that can be invoked through the tool runner can also:

- Accept Unicode UTF-8 strings as option values.
- Be called from an Objectivity/C++ application; see "Invoking Objectivity/DB Tools from Applications" on page 20.

Table 2-1 lists the tools that now have new tool-runner-based versions.

Table 2-1: New Tools Available in Tool Runner

Existing Tool	Corresponding New Tool in the Tool Runner
ooattachdb	AttachDb
oochange	ChangeFd
oochangedb	ChangeDb
oocheckams	CheckAms
oocheckls	CheckLs
oocleanup	CleanupFd
ooconvertformat	ConvertFormat
oocopydb	CopyDb

Table 2-1: New Tools Available in Tool Runner (Continued)

Existing Tool	Corresponding New Tool in the Tool Runner
oocopyfd	CopyFd
oodeletefd	DeleteFd
oodumpcatalog	DumpCatalog
ooexportfd	ExportFd
ooinstallfd	InstallFd
oolicense	License
oolistwait	ListWait
oolockmon	LockMonitor
oonewfd	CreateFd (with the new -noPlacement option)
ooschemadump	SchemaDump
ooschemaupgrade	SchemaUpgrade
ootidy	Tidy

This table is not a complete list of tool-runner-based tools; such a list would also include the placement-management tools introduced in Release 11.0. You can obtain a complete list of tool-runner-based tools as follows:

- At a command prompt, enter:
  ```
  objy ListTools
  ```

NOTE In most cases, each new tool-runner-based tool accepts the same options as its older counterpart, although some differences exist. See *Objectivity/DB Administration* for details about all tools and their options.

Creating Non-Placement-Managed Federated Databases

The `CreateFd` tool now has a new `-noPlacement` option that allows you to create a non-placement-managed federated database. In the previous release, you could use this tool to create placement-managed federations only.

Deprecated Features (Removed in Next Release)

Pre-Release 9.0 Internal Database Format

Release 11.2 is the last release that is backward-compatible with the pre-Release 9.0 internal database format.

NOTE Starting with Release 11.3, new or rebuilt applications will no longer be able to access a pre-Release 9.0 federated database unless it is upgraded to use the Release 9.0 (or later) internal database format.

Background

Objectivity/DB Release 9.0 changed the internal format of system databases and application-created databases, streamlining the implementation of catalogs of databases and containers. The current internal database format improved the speed of lookup operations and doubled the amount of persistent data that can be stored.

The current internal database format is used by:

- All federated databases created with Release 9.0 (or later) of Objectivity/DB.
- Any pre-Release 9.0 federated database that you upgraded using the `ooupgrade` tool.

The older pre-Release 9.0 internal database format is used by:

- Any non-upgraded federated database that was created and populated entirely with pre-Release 9.0 tools and applications.
- Any federated database that was created before Release 9.0, and then partly upgraded, leaving some databases in the pre-Release 9.0 format.
- Any federated database that was created with Release 9.0 (or later), to which databases from a pre-Release 9.0 federated database were attached.

The older format can be accessed by tools and applications built with: Release 9.*x*, Release 10.*x*, Release 11.0, Release 11.1, or Release 11.2. Support for the older format will be discontinued starting with Release 11.3.

What you should do

If you still have any federated databases that use the pre-Release 9.0 internal database format:

➤ Contact Objectivity Customer Support for:

- Help identifying the specific databases to upgrade.
- Assistance with the upgrade procedure itself.

Objectivity/C++

This section describes new, changed, deprecated, and obsolete features of Objectivity/C++.

New and Changed Features

Invoking Objectivity/DB Tools from Applications

You can use the following new classes to invoke selected Objectivity/DB tools from an application:

- `Tool` class
- `ToolParameters` class
- `ToolOutputSink` abstract class and its predefined subclasses:
  - `StandardToolOutputSink`
  - `StringToolOutputSink`
  - `CombinedStringToolOutputSink`

These classes are in the new `objy::tool` namespace.

To use these classes, your application must include `<objy/Tools.h>`. The following directive enables an application to use the classnames without qualification:

```
using namespace objy::tool;
```

You can use these classes to invoke any administration or placement-management tool that is available in the Objectivity/DB tool runner; see "New Tools Available in Tool Runner" on page 17.

Enhanced Navigation Queries

Several enhancement to the navigation-query feature are included with this release.

Enhanced Navigator Components

The following navigator components are enhanced to accept the new navigation-path operators (page 21):

- A result qualifier, which identifies the targets of the navigation query using object qualification, path qualification, or a combination of both.
- A graph view, which provides a mechanism for eliminating uninteresting paths from your navigation query.

Accessing the Source Vertex from a Result Path

In previous releases, access to the source vertex was not available from a result `Path` returned during a navigation query. The `Path` class has a new `relatedObject` member that provides convenient access to the source vertex, which is useful when outputting results in a navigation result handler.

See Chapter 19, "Navigation Queries," in the *Objectivity/C++ Programmer's Guide* for more information.

PQL Enhancements for Navigation Queries

In addition to the new operators listed in "General Predicate Query Language (PQL) Enhancements" on page 14, the following operators are available in Objectivity/C++ navigation queries.

Navigation-Path Operators

New navigation-path operators produce information about a path between related vertices in a graph. Navigation-path operators are used when performing navigation queries.

Operator	Description	Usage	Result Type
`DEPTH` `PATH_LENGTH`	Returns the number of steps in the path.	`DEPTH()` `PATH_LENGTH()`	Unsigned integer
`PREV_EDGE`	Returns the previous edge in the path.	`PREV_EDGE()`	Reference
`PREV_VERTEX`	Returns the previous vertex in the path.	`PREV_VERTEX()`	Reference
`EDGES`	Returns a collection of all edges in the path.	`EDGES()`	Multi-element of reference
`VERTICES`	Returns a collection of all vertices in the path including the starting vertex, but excluding the target vertex.	`VERTICES()`	Multi-element of reference

For example, the following qualifies a path with fewer than three steps.

```
PATH_LENGTH() < 3
```

You can also qualify a path based on its composition. For example, the following qualifies a path where the vertex prior to a designated object (identified elsewhere in the navigation query) is an organization vertex with the name `TPI`.

```
QUALIFY(PREV_VERTEX(), CLASS:Organization, name == 'TPI')
```

See *Objectivity/DB Predicate Query Language* for complete information about navigation path operators and their usage.

PQL Custom Operators

The `evaluate` method of the `objy::query::Operator` class has a new signature.

If you created your own PQL custom operators, you must update these operators to accommodate the changed parameter type. See the *Custom Operator Support* document for 11.2 on the Objectivity Developer Network for more information.

Objectivity for Java

This section describes new, changed, deprecated, and obsolete features of Objectivity for Java.

New and Changed Features

Using Enhanced Predicate Query Language

Objectivity for Java applications can now use the enhanced PQL described in "General Predicate Query Language (PQL) Enhancements" on page 14.

Superseded Features

The following subsections describe superseded Objectivity for Java features.

Named Roots

The named roots mechanism and its supporting methods are superseded by the managed object-placement system introduced in Release 11.0:

- `com.objy.db.app.ooFDObj.bind(Object, String)`
- `com.objy.db.app.ooFDObj.unbind(String)`
- `com.objy.db.app.ooFDObj.lookup(String)`
- `com.objy.db.app.ooFDObj.rootNames()`

An exception is thrown if you attempt to name an object with a root name in a placement-managed federated database.

These items have been noted the *Objectivity for Java Programmer's Reference*, and are still supported for backward compatibility with applications that access existing federated databases.

3

Release Compatibility

This chapter provides information about the impact, if any, of using Release 11.2 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" on page 26 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 28 if you plan to rebuild existing applications with the current release.
- See "Maintaining Earlier Objectivity/DB Applications" on page 29 if you need to continue using unrebuilt tools or applications from an earlier release.

Upgrading Existing Federated Databases

An existing federated database may need upgrading before you can access the data in it with any of the following:

- Tools provided with Objectivity/DB Release 11.2
- New applications built with Objectivity/DB Release 11.2
- Existing applications that have been upgraded, recompiled, and relinked with Objectivity/DB Release 11.2 (see "Upgrading Existing Applications and Scripts" on page 28).

See the following table to determine whether your federated database requires any upgrade.

Federated Database Created With Earlier Objectivity/DB Release	Upgrade Required?	See
Release 11.*x*	*No upgrade required.*	—
Release 9.*x* or Release 10.*x*	Update the Objectivity license	page 26
Release 8.*x* or earlier	Upgrade the internal database format for compatiblity with future releases.	page 27

Updating the Objectivity License

A Release 10.*x* federated database must have an Objectivity Release 11 license to authorize access by Release 11.2 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 11 license.

NOTE An Objectivity Release 11 license authorizes access by Release 11.*x* tools and applications.

To update the Objectivity license for Release 10.x federated databases

1. Verify that you have set up a default license file containing your Objectivity Release 11 license.

   Note: If you did not set up a default Objectivity license file during product installation, see "Setting Up a License File" in Chapter 4 of *Objectivity/DB Administration*.

2. For each Release 10.*x* federated database to be accessed with Release 11.2 tools and applications, enter:

   `objy License -fromdefault -bootFile` *`bootFilePath`*

   where

`-fromdefault`	Obtains `oolicense.txt` in the Objectivity/DB installation directory *`installDir`*.
`-bootFile` *`bootFilePath`*	Path to the boot file of the federated database.

Upgrading Internal Database Format

If you have a federated database that still uses the pre-Release 9.0 internal database format described on page 19, and you plan to continue accessing it in future Objectivity/DB releases, you should upgrade it to use the current internal database format. To do so:

- Contact Objectivity Customer Support for:
  - Help identifying the specific databases to upgrade.
  - Assistance with the upgrade procedure itself.

NOTE If you have already performed the upgrade procedure on a federated database, you do not need to upgrade it again.

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 against the latest headers and relink it with Release 11.2 libraries.

When planning whether to upgrade existing applications to Release 11.2, you should take into account any required code changes listed in the following subsections.

- Upgrading Tool Scripts
- Upgrading Objectivity/C++ Applications
- Upgrading Objectivity for Java Applications

NOTE These subsections describe only the changes introduced in Release 11.2. For descriptions of code and script changes introduced in an earlier release, see the *Objectivity Release Notes* for that release on the Objectivity Developer Network.

Upgrading Tool Scripts

Use the following list to determine whether you must rewrite portions of existing scripts to accommodate Release 11.2 changes to Objectivity/DB tools.

- (Recommended) Review any tool script that uses a tool listed in Table 2-1, and consider updating the script to use the new tool-runner-based tools wherever applicable. The same tool script can invoke any combination of tool-runner-based tools and existing tools, provided each tool uses the correct syntax, tool name, and tool options.

Upgrading Objectivity/C++ Applications

Use the following list to determine whether you must rewrite portions of existing C++ applications to accommodate Release 11.2 changes in Objectivity/C++.

- Adjust any code that defines custom PQL operators to accommodate the change in "PQL Custom Operators" on page 22.

Upgrading Objectivity for Java Applications

Use the following list to determine whether you must rewrite portions of existing Java applications to accommodate Release 11.2 changes in Objectivity for Java.

- If an existing application is to access a placement-managed federated database, remove any code that invokes the superseded named roots mechanism; see "Named Roots" on page 23. You can use the scope naming mechanism instead.

Maintaining Earlier Objectivity/DB Applications

After installing Objectivity/DB Release 11.2 along with your chosen Objectivity programming interface, you normally:

- Develop new Release 11.2 applications.
- Upgrade existing applications and then recompile and relink them with Release 11.2; see "Upgrading Existing Applications and Scripts" on page 28.

In some situations, you may also need to maintain existing applications built with an earlier (pre-Release 11.0) Objectivity/DB release.

Maintaining Unrebuilt Release 10.*x* Applications

Unrebuilt Release 10.*x* applications are not guaranteed to work with either type of Release 11.*x* federated database (placement-managed or non-placement-managed). It is strongly recommended that you upgrade, recompile, and relink existing applications that are to access new Release 11.*x* federated databases.

An unrebuilt Release 10.*x* application can interoperate with new or upgraded Release 11.2 applications only if all applications are accessing existing pre-Release 11.0 federated databases. (The new or upgraded applications must all use the explicit-placement mechanism because existing federated databases are non-placement-managed.)