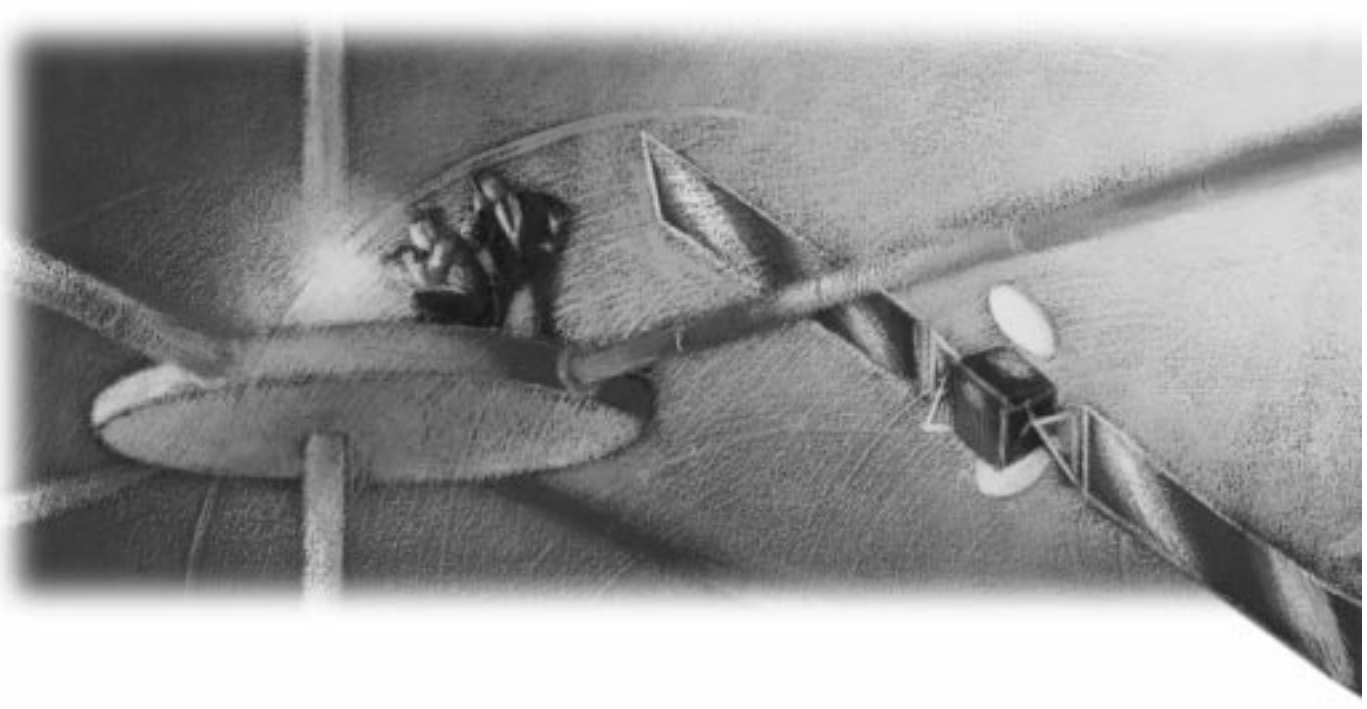

Objectivity Release Notes

Release 6.0

Objectivity Release Notes

Part Number: 60-RN-0

Release 6.0, October 25, 2000

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 1.650.254.7100 *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 1.650.254.7171.
- **Electronic Mail:** Send electronic mail to *help@objectivity.com*.

Before You Call

If you need help from Customer Support, please have the following information ready before you contact Objectivity:

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

1

Release Overview

This release note describes the changes made to Objectivity products and documentation in Release 6.0. This chapter provides an overview of these changes. This chapter summarizes:

- New products
- Updated products
- New and updated books
- Upgrading to this release

NOTE In addition to this printed release note, you should look at the *online release notes*, which describe the supported platforms and compilers, open and fixed software problems, and documentation errata and corrections. You can find the online release notes on your distribution CD and on the Objectivity Technical Support web site. Call Objectivity Customer Support to get access to this web site.

New Products

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

Product Name	Description	See
Objectivity/DB In-Process Lock Server Option (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.	page 17
Rose Objectivity Link	Add-in for Rational Rose C++ products that enables you to generate Objectivity/C++ Data Definition Language files from a Rose model and to create a Rose model from the schema in an Objectivity/DB federated database.	page 23

Updated Products

The following table lists the Objectivity products with new, changed, or deprecated features in this release.

Product Name	Description	See
Objectivity/DB	Distributed object database	page 15
Objectivity/C++	C++ programming interface to Objectivity/DB	page 18
Objectivity/C++ Data Definition Language	Objectivity/C++ option for creating and maintaining a schema of persistence-capable class definitions	page 23
Objectivity/C++ Active Schema	Objectivity/C++ option that enables an application to read and modify a schema dynamically	page 24
Objectivity for Java	Java programming interface to Objectivity/DB	page 26
Objectivity/Smalltalk for VisualWorks	Smalltalk programming interface to Objectivity/DB	page 29

The following table lists the Objectivity products whose features are not changed, but are rebuilt for compatibility with the current release.

Product Name	Description	See
Objectivity/DB Fault Tolerant Option (Objectivity/FTO)	Objectivity/DB option supporting autonomous partitions	page 17
Objectivity/DB Data Replication Option (Objectivity/DRO)	Objectivity/DB option supporting data replication	page 17
Objectivity/DB Open File System (Objectivity/OFS)	Customizable interface between Objectivity/DB and hierarchic storage systems	page 18
Objectivity/C++ Standard Template Library	Objectivity/C++ option that extends ObjectSpace Standards<Toolkit> to add persistence to STL classes.	page 26
Objectivity/SQL++	Server, tools, and programming interface providing ANSI-standard SQL access to Objectivity/DB with object-oriented extensions to SQL	page 31
Objectivity/SQL++ ODBC Driver (Objectivity/ODBC)	Objectivity/SQL++ option that enables ODBC-compliant client applications to access an Objectivity/DB federated database	page 31

New and Updated Books

Printed Books

The following printed books have been updated or rewritten in Release 6.0:

- *Objectivity Release Notes, Release 6* (this document)
- *Installation and Platform Notes for Windows, Release 6*
- *Installation and Platform Notes for UNIX, Release 6*
- *Objectivity/DB Administration, Release 6*
- *Objectivity/FTO and Objectivity/DRO, Release 6*
- *Objectivity/C++ Programmer's Guide, Release 6*
- *Objectivity/C++ Programmer's Reference, Release 6*
- *Objectivity/C++ Data Definition Language, Release 6*
- *Objectivity/C++ Standard Template Library, Release 6*
- *Objectivity/Smalltalk for VisualWorks, Release 6*

Together, *Objectivity/C++ Programmer's Guide* and *Objectivity/C++ Programmer's Reference* replace both *Using Objectivity/C++, Version 4*, and *Objectivity/C++ Supplement, Release 5*.

Online Books

During installation, the online books for Objectivity products are placed in the documentation subdirectory of your Release 6.0 Objectivity / DB installation directory (*`installDir`*).

All Objectivity online books are available on Objectivity's InfoCenter:

`http://info.objy.com`

Books in PDF

The following new or updated books are provided in Portable Document Format (PDF):

- All titles listed under "Printed Books" above
- *Monitoring Lock Server Performance, Release 6*
- *Objectivity/C++ Active Schema, Release 6*
- *Objectivity for Java Guide, Release 6*

PDF files of books containing installation information are located in the root directory of the Objectivity distribution CD.

After you install an Objectivity product, you can access its book(s) from the following PDF file:

- On Windows:
  installDir\doc\ObjyBooks.pdf
- On UNIX (where *arch* is the specific architecture name):
  installDir/arch/doc/ObjyBooks.pdf

You can view the books in PDF using the freely available Acrobat Reader software from Adobe Systems, Inc. You can obtain Acrobat Reader for your platform from Adobe's online services. Use your World Wide Web browser to access the web site www.adobe.com.

Books in HTML

Objectivity for Java Guide and *Objectivity for Java Reference* are available in HTML format. You can use your World Wide Web browser to access these books from the following HTML index file:

- On Windows:
  installDir\doc\java\index.html
- On UNIX (where *arch* is the specific architecture name):
  installDir/arch/doc/java/index.html

Upgrading to This Release

The following subsections provide information about the impact, if any, of:

- Upgrading existing federated databases (see below)
- Upgrading existing applications to the current release (see page 11)

Upgrading Existing Federated Databases

If you created federated databases with earlier releases of Objectivity/DB, you may need to upgrade them to make them compatible with Release 6.0. Depending on the release that was used to create an existing federated database, you may need to upgrade its indexes or schema before it can be accessed by tools or applications built with Release 6.0.

The following table indicates the required upgrade, if any:

You Should Upgrade	In a Federated Database Created With
Indexes	Release 5.0
Schema (persistent collections types)	Any release prior to Release 5.2

Procedures for performing upgrades are in *Installation and Platform Notes* for your platform.

NOTE Objectivity / DB Release 6.0 has the same database format as Releases 4.0.10 or 5.*x*. Consequently, no database-format upgrade is required if you want to access a Release 4.0.10 or 5.*x* federated database using tools or applications built with Objectivity / DB Release 6.0.

Upgrading Existing Applications

You upgrade an existing application by recompiling it and relinking it with Release 6.0 libraries. (Before recompiling an existing Objectivity / C++ application, you must reprocess its DDL files with the Release 6.0 DDL processor.) When planning whether to upgrade existing applications to Release 6.0, you should take into account:

- Lock server considerations
- Required code changes, if any

Lock Server Considerations

Compatible lock-server protocols. Release 6.0 lock servers use a protocol that is compatible with the protocol used by Release 5.2 lock servers. Consequently, applications built with Release 5.2 can use a Release 6.0 lock server, so the same federated database can be accessed concurrently by both a Release 6.0 application and a Release 5.2 application. Existing applications built with Release 5.2 do not need to be upgraded, unless you want to take advantage of Release 6.0 features.

Lock servers with compatible protocols use the same TCP / IP port, so you cannot run both a Release 6.0 lock server and a Release 5.2 lock server on the same workstation at the same time. You should stop the Release 5.2 lock server on each lock server host and run a Release 6.0 lock server instead.

In a partitioned federated database, each autonomous partition can have a different lock server host, and it is possible for some partitions to use Release 5.2 lock servers, while other partitions use Release 6.0 lock servers. If databases are

replicated, however, all partitions containing an image of a given database must use either Release 5.2 lock servers or Release 6.0 lock servers—that is, all partitions replicating the same data must be serviced by the same release's lock server.

Incompatible lock-server protocols. Release 6.0 lock servers use a protocol that is incompatible with the protocol used by lock servers from any release prior to Release 5.2. Consequently, applications built with Release 5.1.*x* or earlier cannot use a Release 6.0 lock server.

Lock servers with incompatible protocols use different TCP / IP ports, so it is possible to run both a Release 6.0 lock server and a lock server from Release 5.1.*x* or earlier on the same workstation. However, if a federated database specifies a lock server host that is running multiple lock servers, you must guarantee that *all* applications accessing a particular federated database have been built with the *same* release of Objectivity / DB (so they will all contact the same lock server).

WARNING Data corruption will occur if two applications contact different lock servers while accessing data in the same federated database.

To prevent data corruption, the same federated database *must not* be accessed concurrently by both a Release 6.0 application and an application built with Release 5.1.*x* or earlier. Therefore:

- If you choose to upgrade an existing application built with Release 5.1.*x* or earlier, you must also upgrade every other application that accesses the same federated database.
- Conversely, if you do not wish to upgrade a particular application built with Release 5.1.*x* or earlier, then no application accessing the same federated database may be upgraded, and any new applications for that federated database must be built with a previous release of Objectivity / DB.

Required Objectivity/C++ Code Changes

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

- If an existing Objectivity / C++ application declares a handle variable as `const`, and that variable is used in an operation that opens the handle, you must change your code to remove `const` from the handle variable declaration. Most Objectivity / C++ functions that open a handle have been changed so that they cannot be performed on a `const` handle. See "Const Keyword Removed from Some Function Prototypes" on page 20.
- If an existing Objectivity / C++ application creates relational operators using constants `ooInt32T` and `ooUint32T` of type `ooDataType`, you must

change these constants to `oocInt64T` and `oocUint64T`. See "Expanded Index and Predicate-Query Capability" on page 19.

- If you added the following symbol definition to a file of a Objectivity / C++ application, you must remove the definition:

  ```
  #define OO_BUGGY_TEMPLATES
  ```

  Objectivity / DB now expects this symbol to be undefined.
- You should consider changing any code that uses features that are deprecated in this release. See "Deprecated Features" on page 21.
- If you have not already done so, you must remove any types and functions that were deprecated in previous releases and removed in this release. See "Obsolete Features" on page 22.

Required Objectivity/C++ Active Schema Code Changes

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

- If an existing Objectivity / C++ Active Schema application uses certain member functions of class `d_Module`, you must change the code to accommodate their new parameters. See "Member Functions of Schema-Description Classes" on page 25.
- If an existing Objectivity / C++ Active Schema application uses certain operators and constructors of class `String_Value`, you must change the code to accommodate their replacements or new parameter types. See "Operators and Constructors of Persistent-Data Classes" on page 25.

  For example, if a statement assigns the result of `Class_Object::get_string()` to a variable of type `ooVString` or `ooVString &`, you must change the code to assign the result to a variable of type `ooVString *`.

Required Java Code Changes

No code changes are required in Objectivity for Java applications.

Required Smalltalk Code Changes

No code changes are required in Objectivity / Smalltalk for VisualWorks applications.

2

New and Changed Features

This chapter describes new and changed features of Objectivity products in Release 6.0.

Objectivity/DB

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

New Features

Lock Server Performance-Monitoring Capability

You can now write a special-purpose program for monitoring the performance of the Objectivity / DB lock server. You can use such a monitoring program to track the interactions between a running lock server and all of the application processes that request services from it. Whereas the `oolockmon` tool provides a snapshot of the locks held at a particular point in time, a monitoring program can be used to obtain very detailed information about lock server usage—for example, when resources are locked and unlocked, when transactions begin and end, when connections with the lock server are made and released, and the order in which lock requests are received from concurrent transactions.

This capability is currently available through a C++ programming interface, which is described in the online book *Monitoring Lock Server Performance*.

Read-Only Databases

You can now designate a database as read-only so that its contents can be read, but not updated, by any application. When a database is read-only, applications can obtain read locks on containers in the database without having to consult the lock server every time such containers are opened. Similarly, all requests for write

locks on the database's containers are automatically denied without requiring repeated interaction with the lock server.

You can set a database as read-only using the `ooattachdb` or `oochangedb` tools. These tools are described in the updated book *Objectivity/DB Administration*.

Changed Features

Performance Improvements

A number of internal changes were made that should improve the runtime speed of your database applications.

Expanded Index and Predicate-Query Capability

A C++, Java, or Smalltalk application can now create indexes and perform lookups with predicate queries that use comparisons to values of fields (data members) whose types are signed or unsigned 64-bit integers.

Programmer-Specified Database Identifiers

You can now specify the numeric identifier for a new database, instead of using the identifier assigned by Objectivity / DB.

You specify a database identifier using the `oonewdb` tool's new `-id` option. This tool is described in the updated book *Objectivity/DB Administration*.

Tool Support for In-Process Lock Servers

The `ookillls` and `oocheckls` tools have been updated to recognize in-process lock servers. See "Objectivity / DB In-Process Lock Server Option" on page 17.

These tools are described in the updated book *Objectivity/DB Administration*.

Options for Controlling Recovery in Partitions

The `oocleanup` tool has two new options (`-onepart` and `-allpart`) that specify whether `oocleanup` should inspect the journal files of one partition or all partitions in a partitioned federated database.

This tool is described in the updated book *Objectivity/DB Administration*.

Objectivity/DB In-Process Lock Server Option

Objectivity / DB In-Process Lock Server Option (Objectivity / IPLS) is a new Objectivity / DB option that enables a C++, Java, or Smalltalk application to start an *in-process lock server*—a lock server that runs as part of the application process. When an application starts and uses an in-process lock server, the application can request locks through simple function calls without having to send these requests to an external process. An in-process lock server can improve the runtime speed of the application that starts it, provided that most or all of the serviced lock requests are from that application.

For complete details, see:

- *Objectivity/DB Administration*
- The books for the various Objectivity programming interfaces

Objectivity/DB Fault Tolerant Option

Objectivity / DB Fault Tolerant Option (Objectivity / FTO) has no new features in this release. See the Technical Support web site for software or documentation problems that have been fixed in this release.

NOTE The C++ programming interface to Objectivity / FTO is now described in the new *Objectivity/C++ Programmer's Guide* and *Objectivity/C++ Programmer's Reference*. Information about the C++ interface has been removed from the *Objectivity/FTO and Objectivity/DRO* book, which describes basic concepts and tools.

Objectivity/DB Data Replication Option

Objectivity / DB Data Replication Option (Objectivity / DRO) has no new features in this release. See the Technical Support web site for software or documentation problems that have been fixed in this release.

NOTE The C++ programming interface to Objectivity / DRO is now described in the new *Objectivity/C++ Programmer's Guide* and *Objectivity/C++ Programmer's Reference*. Information about the C++ interface has been removed from the *Objectivity/FTO and Objectivity/DRO* book, which describes basic concepts and tools.

Objectivity/DB Open File System

Objectivity / DB Open File System (Objectivity / OFS) has no new features in this release. See the Technical Support web site for software or documentation problems that have been fixed in this release.

A new shared library is available for linking to your applications. Contact your account representative to obtain Objectivity / OFS software and documentation.

Objectivity/C++

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

For a complete description of new and changed features, see the new *Objectivity/C++ Programmer's Guide* and *Objectivity/C++ Programmer's Reference*.

New Features

Read-Only Databases

You can now set and manage a read-only database by calling the following new member functions:

- `setReadOnly` member function of the *`ooRefHandle`*`(ooDBObj)` classes
- `isReadOnly` member function of the *`ooRefHandle`*`(ooDBObj)` classes

C++ Interface for Monitoring Lock-Server Performance

You can now create a special-purpose C++ program that monitors the performance of an Objectivity / DB lock server. For a description of the new classes and types, see the online book *Monitoring Lock Server Performance*.

C++ Interface for Objectivity/IPLS

You can now control an in-process lock server in a database application by calling the following new global functions:

- `ooStartInternalLS`
- `ooStopInternalLS`

For more information about in-process lock servers, see "Objectivity / DB In-Process Lock Server Option" on page 17.

Testing a Lock Server

You can now check whether a lock server is running on a particular host by calling the new `ooCheckLS` global function.

Garbage-Collectible Containers

You can now create garbage-collectible containers (for interoperability with Java or Smalltalk applications) by creating instances of the new `ooGCContObj` class.

Changed Features

New Documentation

The features of the Objectivity / C++ programming interface are now described in a pair of new books:

- *Objectivity/C++ Programmer's Guide, Release 6*
- *Objectivity/C++ Programmer's Reference, Release 6*

Together, these books replace Using *Objectivity/C++, Release 4*, and *Objectivity/C++ Supplement, Release 5*.

Programmer-Specified Database Identifiers

You can now set a database's identifier by specifying the new *`id`* parameter of the `ooDBObj` constructor.

Expanded Index and Predicate-Query Capability

Indexes and predicate queries now support lookup of 64-bit integer data members.

You can now define custom relational operators that provide return values for 64-bit operands. To do so, you use the following new constants of the `ooDataType` global type:

- `oocInt64T` (replaces constant `oocInt32T`)
- `oocUint64T` (replaces constant `oocUint32T`).

Const Keyword Removed from Some Function Prototypes

You can no longer use `const` handle variables in operations that open a handle. Consequently, the `const` keyword has been removed from the prototypes of functions that open a handle:

- In the `ooDelete` and `ooDeleteNoProp` global functions, parameters of type `ooHandle(`*`className`*`)` are no longer `const`.
- Member functions of handle classes are no longer `const` if they open the handle on which they are called. The changed member functions in each handle class are shown in Table 2-1.

Table 2-1: Member Functions That are No Longer `const`

ooHandle(ooObj)	ooHandle(ooContObj)	ooHandle(*appClass*)	ooHandle(ooAPObj) ooHandle(ooDBObj) ooHandl(ooFDObj)

`open`
`ptr`
`update`
 	
`operator->`
`operator*`
`operator ooObj*`
	`open`
	`ptr`
	`refreshOpen`
	`update`
	
	`operator->`
	`operator*`
	`operator ooContObj*`
		`open`
		`ptr`
		`refreshOpen`
		`update`
		
		`operator->`
		`operator*`
		`operator appClass*`
			`open`
			`update`

Transaction Identifier

You can now obtain the integer identifier of a transaction by calling the `getId` member function of class `ooTrans`. Administration tools such as `oolockmon` and `oolistwait` refer to a transaction using its identifier.

Persistent Collections

You can now specify the initial number and distribution of hash buckets by using new constructors of class `ooHashSet` and class `ooHashMap`.

While iterating over keys in an object map, you can now find the value that is paired with the current key by calling the `currentValue` member function of class `ooCollectionIterator`.

You can now refresh the containers that are used internally by a collection by calling the `refresh` member function of class `ooCollection`.

Template Implementation of Class ooString(*N*)

The name `ooString(N)` is now a macro that expands to a template class whose parameter is *N*. Consequently, you can now use the name without having to declare and implement it with the C++ `declare` and `implement` macros.

ooExitCleanup Usage

The `ooExitCleanup` global function is now necessary only in multithreaded Objectivity / C++ applications running on Windows platforms.

oo_vc_version Environment Variable on Windows

You no longer need to set the `oo_vc_version` environment variable on Windows platforms. Objectivity / C++ uses the Microsoft Visual C++ Release 6.0 compiler by default, and the Visual C++ Release 5.0 compiler is no longer supported.

NOTE If you set this environment variable for the previous Objectivity / C++ release, you must make sure that the variable currently has either the value `6.0` or no value; alternatively, you can delete the environment variable.

Change to Link Options on Solaris 7

When you link an Objectivity / C++ application with the Objectivity / DB shared library on Solaris 7, you no longer have to include the following option in your link rule:

```
-library=iostream,no%Cstd
```

Link rules for other architectures have not changed. Linking is described in the *Installation and Platform Notes for UNIX* and the *Installation and Platform Notes for Windows*.

Deprecated Features

Deprecated features will be removed from the Objectivity / C++ interface in the next release.

Reserve Locks

You should not use the functions for checking out objects and checking them back in under a user ID. These functions have been removed from the documentation:

- `checkin` member function of the *`ooRefHandle`*`(ooObj)` class
- `checkout` member function of the *`ooRefHandle`*`(ooObj)` class

Handle Stack

You should not use the types and functions for tracking active handles on a handle stack, because this feature is no longer compatible with the current handle implementation. The following types and functions have been removed from the documentation:

- `ooMark` global type
- `ooSetMark` global function
- `ooReleaseMark` global function

Keyed Objects

You should not use the types and functions for creating and finding keyed objects. Instead, you should use the `ooHashSet` class with an application-defined comparator to provide fast content-based lookup of objects in a hash table. The following types and functions have been removed from the *Objectivity/C++ Programmer's Guide*, but still appear in the *Objectivity/C++ Programmer's Reference*:

- `ooKey` global type
- `ooKeyType` global type
- `ooNewKey` global function
- `ooGetMemberOffset` global function
- `ooGetMemberSize` global function
- Overloading of *`ooRefHandle`*`(ooObj)::lookup` that accepts a keyed object as a parameter value.

Internal Function for Getting a Transaction Identifier

The following undocumented function has been replaced by the new `getId` member function on the `ooTrans` class:

- `getInternalId` member function of the `ooTrans` class.

Obsolete Features

Obsolete features are removed from the Objectivity / C++ interface in this release.

oovTopFD and oovTopDB Scope

Overloadings of the following member functions were removed if they used `oovTopFD` or `oovTopDB` as an implicit default scope:

- *`ooRefHandle`*`(ooObj)::getObjName, lookupObj, nameObj, unnameObj`
- *`ooRefHandle`*`(ooContObj)::exist, lookupObj, open`

- *ooRefHandle*(ooDBObj)::exist, open
- *ooRefHandle*(ooAPObj)::open

Objectivity/C++ Data Definition Language

This section describes new, changed, and obsolete features of Objectivity / C++ Data Definition Language (Objectivity / DDL) in this release. See the Technical Support web site for software or documentation problems that have been fixed in this release.

Changed Features

C++ Parser

The C++ parser component of the DDL processor has been upgraded to more closely conform to the C++ standard and to more fully support platform-dependent language extensions. Consequently, the DDL processor now accepts most modern C++ language constructs in a DDL file without reporting them as syntax errors. For example, the DDL processor now accepts `long double` literals.

Rose Objectivity Link

Rose Objectivity Link is a new Objectivity product that enables you to employ Rational Rose C++ products when developing a federated-database schema for an Objectivity / C++ application. With Rose Objectivity Link, you can use Rational Rose 98i or Rational Rose 2000 to generate Objectivity / C++ Data Definition Language files from a UML model and to extract a UML model from an existing federated-database schema.

Contact your account representative to obtain Rose Objectivity Link software and documentation.

Objectivity/C++ Active Schema

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

For a complete description of new and changed features, see the updated *Objectivity/C++ Active Schema* book.

New Features

New Member Functions in Schema-Description Classes

Class `d_Class` now has new member functions:

- `get_static_ref` member function
- `set_static_ref` member function

Class `d_Module` now has new member functions:

- `propose_versioned_class` member function
- New overloading of `propose_new_class` member function:
  `Proposed_Class propose_new_class(Proposed_Class *newClass)`

Class `d_Scope` now has new member functions:

- `is_class` member function
- `is_module` member function

New Member Functions in Proposal-Descriptor Classes

Class `Proposed_Class` now has the following new member functions and constructor:

- `position_in_class` member function
- `add_property` member function:

```
ooStatus add_property (int32 position,
                       d_Access_Kind visibility,
                       const d_Property &existingProperty);
```

- `Proposed_Class(const char *name, ooTypeNumber tnum)`

New Classes and Member Functions for Exceptions

New exception classes:

- Class `ProposeEvolAndVers`, with member function `class_name`
- Class `DeletedClassObjectDependency`, with member function `persistent_data_object_of`

Class `ModuleInitError` is now an exception class, because it now derives from `asException` instead of `asError`.

New member functions:

- `mode` member function in `FailedToOpenObject` class
- `mode` member function in `FailedToReopenFD` class

Changed Features

Member Functions of Schema-Description Classes

In class `d_Module`, each of the following member functions now has a new parameter of type `ooHandle(ooFDObj) &` for accepting a handle to the current federated database:

- `activate_remote_schema_changes` member function
- `sanitize` member function

Operators and Constructors of Persistent-Data Classes

In class `String_Value`, the operators have been replaced as shown:

This Operator	Now Replaced With This Operator
`operator ooVString`	`operator ooVString *`
`operator ooUtf8String`	`operator ooUtf8String *`
`operator ooSTString`	`operator ooSTString *`

Various `Class_Object` constructors now have parameters of type `ooHandle(ooObj) &` instead of `const ooHandle(ooObj) &`.

Postfix operator ++ in Iterator Classes

In the following iterator classes, the postfix `operator++` now returns a *copy* of the iterator being used, *not* a reference to that iterator:

- `list_iterator` class
- `meta_object_iterator` class

- `type_iterator` class
- `attribute_plus_inherited_iterator` class
- `base_class_plus_inherited_iterator` class

Objectivity/C++ Standard Template Library

Objectivity / C++ Standard Template Library has no new features in this release. See the Technical Support web site for software or documentation problems that have been fixed in this release.

Obsolete Features

Objectivity is evaluating whether to discontinue support for nondefault allocators in Objectivity / C++ STL containers. For compatibility with future releases of Objectivity / C++ STL, your applications should use only the default allocator for each Objectivity / C++ STL container. This allocator clusters the elements of the Objectivity / C++ STL container within a single Objectivity / DB container. If your Objectivity / C++ STL containers become so large that you need to distribute elements among multiple Objectivity / DB containers, you should consider using an Objectivity / C++ scalable persistent collection instead.

Objectivity for Java

This section describes new, changed, 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.

For a complete description of new and changed features, see the updated *Objectivity for Java Guide* and *Objectivity for Java Reference*.

New Features

Objectivity for Java has new methods (but no new packages, classes, or interfaces) in this release.

Read-Only Databases

You can now set and manage a read-only database by calling the following new methods:

- `com.objy.db.app.ooDBObj.setReadOnly`
- `com.objy.db.app.ooDBObj.isReadOnly`

Java Interface for Objectivity/IPLS

You can now control an in-process lock server in a database application by calling the following new methods:

- `com.objy.db.app.Connection.startInternalLS`
- `com.objy.db.app.Connection.stopInternalLS`

For more information about in-process lock servers, see “Objectivity / DB In-Process Lock Server Option” on page 17.

Testing a Lock Server

You can now check whether a lock server is running on a particular host by calling the following new method:

- `com.objy.db.app.Connection.checkLS`

Session Control

You can now install a custom signal handler in a session by calling the following new method:

- `com.objy.db.app.Connection.setInstallSignalHandler`

You can now specify how long a session is to wait for an Objectivity server to respond before signaling a timeout error. To do so, you call the following new method:

- `com.objy.db.app.Session.setRPCTimeout`

Deployment Mode

You can now enable and manage deployment mode by calling the following new methods:

- `com.objy.db.app.Connection.setDeploymentMode`
- `com.objy.db.app.Connection.isDeploymentMode`

When deployment mode is enabled, Objectivity for Java does not check each persistence-capable class definition to distinguish new classes from modified existing class. Enabling deployment mode improves application performance because it allows Objectivity for Java to bypass schema comparisons that are performed by default. You should enable deployment mode in an application only if you are certain that no schema evolution is required.

Cache Control

You can now control when and how objects are removed from the cache by calling the following new methods:

- `com.objy.db.app.ooObj.dropCachedReference`
- `com.objy.db.app.Session.setFlushCacheAfterCommit`
- `com.objy.db.app.Session.setFlushCacheAndDeadenObjectsAfterCommit`

Changed Features

Programmer-Specified Database Identifiers

You can now specify a database's identifier by calling the new variant of the following method:

- `com.objy.db.app.ooFDObj.newDB`

Expanded Index and Predicate-Query Capability

Indexes and predicate queries now support lookup of 64-bit integer fields.

Persistent Collections

You can now specify the initial number and distribution of hash buckets by using new constructors of the following classes:

- Class `com.objy.db.util.ooHashMap`
- Class `com.objy.db.util.ooHashSet`

While iterating over keys in an object map, you can now find the value that is paired with the current key by calling the following new method:

- `com.objy.db.util.CollectionIterator.currentValue`

You can now refresh the containers that are used internally by a collection by calling the following new method:

- `com.objy.db.util.Collection.refresh`

Retrieving Objects by OID

You can now specify a lock mode when you retrieve a persistent object with a specified object identifier (OID). To do so, you call the new variant of the following method:

- `com.objy.db.app.ooFDObj.objectFrom`

Container Lookup

You can now distinguish between a non-existent or a locked container when performing a container lookup using the following method:

- `com.objy.db.app.ooDBObj.lookupContainer`

Instead of simply throwing `ObjyRuntimeException` for both conditions, the method now throws either `ObjectNotFoundException` or `LockNotGrantedException`. Both of these new exceptions are subclasses of `ObjyRuntimeException`, so you do not have to change existing code unless you want to take advantage of this distinction.

Objectivity/Smalltalk for VisualWorks

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

For a complete description of new and changed features, see the updated *Objectivity/Smalltalk for VisualWorks* book.

New Features

Read-Only Databases

You can now set and manage a read-only database by sending the following new messages:

- `aDB setReadOnly: aBoolean`
- `aDB isReadOnly: aBoolean`

Smalltalk Interface for Objectivity/IPLS

You can now control an in-process lock server in a database application by sending the following new messages:

- `OoSession startInternalLS`
- `OoSession stopInternalLS`
- `OoSession stopInternalLS: wait force: aBoolean`

For more information about in-process lock servers, see “Objectivity / DB In-Process Lock Server Option” on page 17.

Testing a Lock Server

You can now check whether a lock server is running on the current or specified host by sending the following new messages:

- `OoSession checkLS`
- `OoSession checkLS: aHostName`

Container Size

You can now obtain the number of logical pages and physical (storage) pages in a container by sending the following new messages:

- `aContainer numLogicalPages`
- `aContainer numPhysicalPages`

Hot Mode

You can now enable hot mode, which can improve performance when data is created and saved by applications on different platforms. To do so, you send the following new message:

- `aSession hotMode: aBoolean`

Changed Features

Programmer-Specified Database Identifiers

You can now set a database's identifier by sending the following new message:

- ```
  aSession newDB: aString defaultContPages: pages
  growth: percent host: hostName path: path weight: weight
  userDBID: userDBID
  ```

Expanded Index and Predicate-Query Capability

Indexes and predicate queries now support lookup of 64-bit integer instance variables.

Persistent Collections

You can now specify the initial number and distribution of hash buckets by sending the following new messages:

- `aHashSet bucketContainer: aContainer`
- `aHashSet initialBuckets: anInteger`
  ```

While iterating over keys in an object map, you can now find the value that is paired with the current key by sending the following new message:

- `aCollectionIterator currentValue: openMode`

You can now refresh the containers that are used internally by a collection by sending the following new message:

- `aCollection refresh: openMode`

The method `adminContainer: aContainer` has been moved from the `OoBTree` class to the `OoCollection` class.

Objectivity/SQL++

Objectivity / SQL++ has no new features in this release. See the Technical Support web site for software or documentation problems that have been fixed in this release.

Objectivity/SQL++ ODBC Driver

Objectivity / SQL++ ODBC Driver (Objectivity / ODBC) has no new features in this release. See the Technical Support web site for software or documentation problems that have been fixed in this release.

OBJECTIVITY, INC.
301B East Evelyn Avenue
Mountain View, California 94041
USA
+1 650-254-7100
+1 650-254-7171 Fax
www.objectivity.com
info@objectivity.com