Installing ObjectStore for UNIX

Release 6.1
February 2003

Installing ObjectStore for UNIX

ObjectStore Release 6.1 for all platforms, February 2003

Contents

Chapter 1 **Overview 1**
ObjectStore Components 1
ObjectStore DBMS 2
C++ Interface 2
Java Interface 2
Javlin 3
Dynamic Data Modeling Language 3
ObjectStore Platform Configurations 3
Types of Installation Procedures 4
New Installation 4
Upgrade from Release 6.0 4
Upgrade from 5.1 Service Pack x 5
Rolling Upgrade 5
Adding a Component 6
Installation for a Sun Cluster Environment 6
Configuration 6
Installation Concepts 7
ObjectStore Process Architecture 7
Development Configuration 9
Runtime Configuration 9
Types of Databases 9
What is the Transaction Log? 10
What is a RAWFS? 11
Overview of the Installation Process 11
Getting Help from Technical Support 13

Chapter 2 **System Requirements for ObjectStore 15**
Supported Platforms 15
Hardware Requirements 15
CPU 15
Memory 15

Disk Space for the Application and ObjectStore 16
Software Requirements 17
Installation Media for ObjectStore 18
Obtaining Installation Media 18
Description of ObjectStore Releases 19

Chapter 3 Installing and Configuring ObjectStore 21
Installing ObjectStore for the First Time 21
Preparing for Installation 21
Installation Procedure 22
ObjectStore Environment Variables 23
Verify Installation 24
Verify Installation Directory Contents 24
Verify a C++ Development Client 26
Verify a C++ Runtime Client 26
Verify a Java Client 27
Upgrading from ObjectStore 6.0 Service Pack x 28
Upgrading from ObjectStore 5.1 Service Pack x 29
Installing and Configuring for a Rolling Upgrade 30
Configuration Tasks 32
Configuring Host for Client and Server using File Databases (no RAWFS) 33
Configuring Host for Client and Server with RAWFS or File Database capability 34
Configuring Host for Client Only 36
Configuring ObjectStore for Failover 37
Configuring for Java Interface Use on This Host 37
Installing and Configuring ObjectStore on Sun Clusters 39
Using a Rolling Upgrade to Install and Configure ObjectStore on Sun Clusters 41
Troubleshooting Installation Problems 43

Chapter 4 Miscellaneous Information 45
Uninstalling ObjectStore 45
Adding a Component 46

Resource Group Properties for Sun Clusters 46
Resource Properties for Sun Clusters 47
ObjectStore-Specific Resource Properties 49
Location of Schema Databases 50

Chapter 1
Overview

ObjectStore is an industrial-strength Object Database Management System (DBMS) that enables applications to store and manage any type of data, no matter what the complexity. It features a distributed process architecture that allows for applications to scale and enables in-memory access to data of any type (C++, Java, XML). The ObjectStore DBMS supports any variety of distributed processing through its flexible process architecture which ensures transactional consistency in a wide variety of network configurations. The ObjectStore server process supports clients on many different platforms; see the Support Matrix on the Technical Support web site (http://www.objectstore.net/support/matrix) for the most up to date information.

ObjectStore applications running on different platforms can all access the same data. ObjectStore has all of the necessary features for system management which include on-line backup and restore, archive logging and recovery, replication, transparent failover and many other system management operations necessary in an enterprise DBMS.

ObjectStore Components

ObjectStore contains several components that can be used in a development or in a deployment environment. This document refers to the latter as a *runtime environment*. The type of ObjectStore installation depends on the type of environment required. Here are the major components of an ObjectStore environment.

ObjectStore DBMS

The ObjectStore DBMS is the database server and associated files that make up the core ObjectStore services that can serve both C++ and Java applications. This document refers to the DBMS as the *server* because in an ObjectStore application, the ObjectStore server communicates with any type of application (C++ or Java) and accesses ObjectStore databases on behalf of the C++ or Java client applications. Along with the ObjectStore server, the DBMS also includes a number of utilities and libraries that can be used to administer databases in an ObjectStore environment. Which utilities and libraries are installed will depend upon whether the environment is for C++ or Java.

C++ Interface

C++ developers write client applications using the ObjectStore C++ Application Programming Interface (API) to access ObjectStore databases in a client process that communicates with the ObjectStore server process. Developers access data in their databases as they would any C++ object in program memory, but use the ObjectStore C++ API for controlling placement of their objects in the database and for transactional control. ObjectStore includes a collections library that provides application developers the ability to aggregate, query, and index C++ objects in a database. ObjectStore supports C++ applications written on different platforms and compilers, all accessing the same data in a distributed environment. The C++ interface includes the C++ Middle-Tier Library (CMTL) that allows the C++ developer to develop applications with distributed caches.

Java Interface

Java developers write client applications using the ObjectStore Java API to access ObjectStore databases in a Java Virtual Machine (JVM) which communicates with the ObjectStore server process.

Developers access data in their databases as they would any Java object in program memory, but use the ObjectStore Java API for controlling placement of their objects in the database and for transactional control. The JDK Collections and ObjectStore Collections classes provide application developers the ability to aggregate, query, and index Java objects in a database. ObjectStore supports Java applications written on different platforms, all accessing the same data in a distributed environment.

Javlin

The Javlin component is used to integrate with several application servers for either EJB or J2EE JTA transaction support (or both). Javlin includes the Java Middle-tier Library (JMTL) that developers use when they want to write middle-tier applications that utilize the power of transactionally-consistent distributed data caches. The Javlin component utilizes the Java interface to ObjectStore, along with a specialized set of classes which enable the batching and routing of transactional activity to be sent to the ObjectStore server process.

Dynamic Data Modeling Language

The Dynamic Data Modeling Language (DDML) is a class library which allows for an easy-to-use model for the creation of name/value pairs for flexible schema data access in an ODBMS. With the C++ interface, developers can use DDML to extend schemas at run-time.

ObjectStore Platform Configurations

ObjectStore is a product that may be installed on multiple types of hardware, operating systems, and in conjunction with a variety of compilers, language-related libraries, JDKs and JREs. The product is built to run on a set of platforms.

ObjectStore supports a number of platforms, including Windows, Solaris, and Linux. For each of these platforms, there are a number of compilers or JDKs that the product must be built with in order for the product to work for an application. In this document, *platform configuration* refers to a combination of compiler and operating system. In order to install ObjectStore, it is necessary to understand what platform configuration you require for your application.

Supported platform configurations can be found in `ReleaseNotes.pdf` and in the Support Matrix on the Technical Support web site (http://www.objectstore.net/support/matrix).

Types of Installation Procedures

The procedures that you follow in order to install this release of ObjectStore or any of its components will depend on what your goal is. Here is a summary of the ways in which the ObjectStore installation procedures may be used on a given platform.

New Installation

This is the most straightforward type of installation because there is no existing application or set of databases that need to be migrated. These procedures should be followed when ObjectStore is being installed for the first time for new development or for setting up a system for deployment.

Upgrade from Release 6.0

Upgrading from a previous 6.0 release is very straight-forward, unless the application is being built using a different compiler than was used in 6.0. You should first make sure that the platform configuration is supported for this release and service pack; see the Support Matrix on the Technical Support web site (http://www.objectstore.net/support/matrix). An upgrade will require that you first ensure that all applications running on a given host that use the ObjectStore server process are no longer running, the ObjectStore server process has checkpointed the databases it has open (propagate transaction log to physical databases) and that the necessary ObjectStore processes have been cleanly shut down. The checkpointing and shutting down of the ObjectStore server process is only necessary if you are upgrading the host where the ObjectStore server process is running.

When 6.1 is installed, the installation process will ask if you want to overwrite the existing ObjectStore binaries, delete and install them, or install them in a new location. It will be the responsibility of the user doing the installation to make sure that all databases are properly backed up prior to the upgrade. For an upgrade from 6.0 to 6.1, application developers will need to recompile and relink their applications. In general, databases created in 6.0 will be compatible for 6.1 applications. The exceptions to this will be if there is a major compiler change (see below). The rules concerning compatability of ObjectStore processes in a mixed-release environment will hold. That is, a 6.0 ObjectStore server process will not be able to communicate with a 6.1 ObjectStore client process but a 6.1 ObjectStore server process will be able to communicate with a 6.0, 6.1, or even a 5.1 ObjectStore client process. This means that if you are upgrading a number of host systems in a distributed

environment, it is important that you first upgrade the hosts where the ObjectStore server process reside.

In addition to database compatibility between releases, there are some code changes that users will need to make if they are using deprecated APIs. The deprecated APIs are detailed in the Release Notes for this release.

ObjectStore does not automatically support an upgrade of an application whereby the compiler is changing. Please refer to the Migration Guide for instructions for how to upgrade to a different compiler. Examples of common compiler changes as customers upgrade to this release include Sunpro 4.2 to Sun ONE Studio 7 on Solaris and VC6 to VC7 on Windows.

As more platforms/compilers are available for ObjectStore 6.1, more options will be available for developers to upgrade their ObjectStore environments.

A new feature of ObjectStore 6.1 is the ability to perform a *rolling upgrade* — that is, upgrade an ObjectStore server process without having to shut down client applications. You must have confiured ObjectStore for failover inorder to use this feature. For more information, see Installing and Configuring for a Rolling Upgrade on page 30.

Upgrade from 5.1 Service Pack x

This procedure is used when you have an existing 5.1.x release of ObjectStore and you want to upgrade to 6.1. This type of installation requires that you perform a database migration and application source code migration in order for you to prepare to use release 6.1 You should consult the Support Matrix on the Technical Support web site (http://www.objectstore.net/support/matrix) for the platform configurations that are supported for this release of ObjectStore. There are some platform configurations that used to be supported in release 5.1 which are no longer supported. In order to upgrade from 5.x to 6.1, you should follow the detailed instructions in the Migration Guide, which can be found at http://www.objectstore.net/documenation/migration.

Rolling Upgrade

If you have configured ObjectStore to support failover, you can upgrade ObjectStore from 6.0 to 6.1 by performing a rolling upgrade. In a rolling upgrade, ObjectStore is installed and configured to run in a mode whereby the ObjectStore Server process is upgraded to the new release first and the clients can continue running. A rolling upgrade uses the failover mechanism to shut down the existing ObjectStore server processes and reconnect the

ObjectStore clients to the new ObjectStore server process without interruption in service. In a rolling upgrade scenario, you upgrade (via `osconfig`) the ObjectStore Server processes (each one participating in the failover configuration) and then when it is a convenient time, upgrade the clients (via `osconfig`). For more information, see Installing and Configuring for a Rolling Upgrade on page 30.

Adding a Component

This procedure is used when you have installed some components of the ObjectStore product suite but need to install more components. An example of this would be if you installed the Java Interface and now want to install the Javlin component.

Installation for a Sun Cluster Environment

A customer who already possesses a Sun Clusters 3.0 system and has a good understanding of how to manage that system should be able to install ObjectStore onto it and achieve high availability for databases. This comes from a combination of storing databases and the transaction log on mirrored disks, and providing failover support for failures of the osserver process or the node on which it is running. The Sun Clusters system provides the mirroring and failover. ObjectStore provides the software needed to fit into this system.

Configuration

On UNIX, the configuration is performed as separate step after the installation is complete using a separate process called `osconfig`. The types of post-installation configuration that can be done are:

- Configure host as a server. Note that clients can also run on a host configured as a server.
- Configure host as a client.
- Configure host for Sun Clusters.
- Configure host for failover.

See Configuration Tasks on page 32 for more details.

Installation Concepts

ObjectStore Process Architecture

The main processes in an ObjectStore environment depend upon the components installed and whether the application is a Java application or a C++ application. The core environment for ObjectStore is C++. If your application is written in Java, you will be executing some C++ code at the storage management level automatically. That is why the C library is required in an ObjectStore environment.

All access to ObjectStore databases is done via the ObjectStore server process (`osserver`). The ObjectStore server process must reside on the same host where the physical databases reside. This eliminates the need for the ObjectStore server process to access databases over the network.

On the *server* side, the resources are: the ObjectStore server process, any databases that it is accessing, and the ObjectStore transaction log file which also must reside on the same host as the ObjectStore server process. There is always one and only one ObjectStore transaction log file, no matter how many databases the ObjectStore server process is managing. A database can only be managed by a single ObjectStore server process. The ObjectStore transaction log is a special file that holds all transaction data that needs to be propagated to the databases that the ObjectStore server process is managing. The ObjectStore transaction log is a critical file that should never be removed unless you are sure that all data has been propagated to the databases.

Any number of ObjectStore clients can access databases via one or more ObjectStore server processes. (Each ObjectStore server process needs to be running on a separate host). In order for an ObjectStore client to access a database, it communicates with two processes: the ObjectStore server process and the ObjectStore cache manager process (`oscmgr6`). The ObjectStore server process grants access to clients and the ObjectStore cache manager process keeps track of locks that all clients have on a particular host with any number of ObjectStore server processes. The ObjectStore client can either be a C++ process that is linked with the ObjectStore libraries or a JVM (Java Virtual Machine) that utilizes the ObjectStore classes in the ObjectStore `.jar` files.

When Javlin is installed and configured, either for development or runtime, the process architecture involves the use of an application server. There are many design choices that can be made in the use of Javlin and how it will utilize an application server. A typical configuration would be one where

multiple JVMs are instantiated, each of which could be an ObjectStore client process.

The following figure illustrates a simple ObjectStore process architecture:

In this environment, there is a single ObjectStore server process and multiple ObjectStore client processes, all running on the same host. This environment requires a development or runtime installation, which includes a C++ environment. If you require OSJI, Javlin, or DDML, then you need to install those components. This host would be configured as a server.

The following figure illustrates a multi-host ObjectStore process architecture involving a single ObjectStore server process:

In this environment, there is a single ObjectStore server process and multiple ObjectStore client processes. The ObjectStore server process has been set up on Host1 which is equipped with perhaps a larger set of disks for databases. The ObjectStore clients can run on either Host1 or Host2. For this configuration, you will install ObjectStore (runtime or development) on each host. If you require OSJI, Javlin, or DDML, make sure to install them. Host1 is configured as a server and Host2 is configured as a client.

The following figure illustrates a multi-host ObjectStore process architecture involving multiple ObjectStore server processes:

In this environment, there are multiple ObjectStore server processes running on multiple hosts and multiple ObjectStore client processes. The ObjectStore server processes have been set up on both hosts. Both Host1 and Host2 must have adequate disk space to support locally accessible ObjectStore databases. The ObjectStore clients can run on either Host1 or Host2. This environment requires runtime or development installation. You can install OSJI, Javlin, or DDML. Both Host1 and Host2 will be configured as servers.

The following figure illustrates a very simple process architecture where Javlin is installed, along with an application server

In this environment, there is a single ObjectStore server process and multiple ObjectStore client processes. Some clients are running on Host1 and on Host 2, there is an application server which is using Javlin. The ObjectStore clients can run on either Host1 or Host2. This environment requires a runtime or development installation. Host1 may have OSJI, Javlin, and DDML installed on it. Host2 must have OSJI and Javlin installed on it (DDML is optional). Host1 will be configured as a server and Host2 will be configured as a client.

Development Configuration

A development configuration typically involves the installation of all ObjectStore components for development and runtime along with the appropriate compilers or JDK. C++ developers need access to an ObjectStore server process in order to build the application. This is because the C++ build process involves the creation of schema that must be created using the ObjectStore server process. A developer needs access to the C++ include files and libraries or the Java `.jar` files in order to build and test the application. There are special utilities installed in a development environment that allow schema to be created. Any number of developers can share the resources of an ObjectStore server process on a *shared* host, provided that the developers who need to access databases (during build or testing) have access to a host with an ObjectStore server process and the databases used by the application for building or testing will be local to that host where the ObjectStore server process is running.

Runtime Configuration

A run-time or deployment configuration should be based on the application needs. It is used primarily for a deployed application because it does not contain any of the schema generation utilities needed to build an application. A deployment configuration may involve a dedicated host that runs only the ObjectStore server process. All clients run on different hosts. It may involve a pool of hosts that will have ObjectStore server processes on them. The ObjectStore server process hosts will need to be capable of having very large disk storage capabilities. Memory requirements are not as important for the ObjectStore server process.

The ObjectStore client processes will require the most memory. Each client will need a minimum amount of memory and backing disk store for paging. How many clients one configures to run per host will depend on the application design.

It is certainly possible to have both the ObjectStore server process and the ObjectStore client process(es) running on the same host. This is an application design choice.

Types of Databases

An ObjectStore database can be a file in the operating system file system. The exception to this is a RAWFS database; see What is a RAWFS? on page 11. The only ObjectStore process which physically accesses the database is the ObjectStore server process. A database contains schema that describes the

types of objects that this database can contain. Only instances of objects of the defined types may be stored in the database. An ObjectStore database can only be accessed or viewed by an application or specialized tools and utilities that are developed to access the data in the database. An ObjectStore database can only be served by a single ObjectStore server process. An application can access any number of ObjectStore databases at the same time. The physical database files or partitions must reside on the local file system of the host where the ObjectStore server process is running. A special type database with the suffix of `.adb` and `.ldb` is used by applications in order to ensure that the schema of objects being used by the client applications match the schema of the database containing the data. The special databases are called application schema databases and library schema databases. Since these special files are databases, they can only be accessed by the ObjectStore server process. These databases must be placed on a host file system where an ObjectStore server process is running.

What is the Transaction Log?

The ObjectStore transaction log is a special file that is only accessed by the ObjectStore server process. It should be configured to reside on the same host where the ObjectStore server process is running. In addition, for performance reasons, it is recommended that the ObjectStore transaction log is configured to be located on a different physical disk drive than the disk drive which will hold the databases that the ObjectStore server process will access. Also, the transaction log should be on a dedicated disk drive that is not used for any other purpose.

The ObjectStore transaction log is the file that contains the *active* and some committed transactions for all databases that the ObjectStore server process is serving to clients. At frequent intervals, the contents of the log file for committed transactions is propagated to the databases. For this reason it is extremely important that the ObjectStore transaction log not be removed unless a special configuration change is being made and the system administrator has ensured that all transaction data has been forced to be propagated to the databases. This forcing of the propagation can be achieved by checkpointing the ObjectStore server process and by shutting down the ObjectStore server process. In order for this effective checkpointing or shutdown to occur, it is also necessary that the system administrator can ensure that no ObjectStore client applications access the databases served by this ObjectStore server process. Once the ObjectStore server process is completely shut down, the ObjectStore transaction log can be safely removed. This enables the system administrator to initialize a new

ObjectStore transaction log file either in the same location or in a different location.

What is a RAWFS?

RAWFS is the term for the ObjectStore RAW File System. This is a special area or partition on a disk that is reserved for use only by the ObjectStore server process. Only one ObjectStore server process may access a RAWFS. One advantage of a RAWFS is that it can span multiple disk partitions, allowing for very large databases. Another advantage is that the files (databases) and directory structure that make up the RAWFS are not visible from the operating system using normal commands (`dir` or `ls`). This special file system is only visible via the ObjectStore commands or by an ObjectStore client process.

A RAWFS is expandable, but it cannot be made smaller. If you want a RAWFS to be smaller, you have to back up the databases in the RAWFS, remove the RAWFS and recreate a smaller one.

If you are planning on using a RAWFS partition, you should make sure the disk partition has been already created prior to the configuration of the partition to be used by ObjectStore. If you are planning on using a file for a RAWFS, you should make sure that there is adequate disk space for the RAWFS that you want it to contain. Finally, a RAWFS for a host must be local to that host.

Overview of the Installation Process

1. Determine which platform configuration that you need to install and obtain the installation media either on CD or by downloading from the Technical Support web site.
2. Read the `README.htm` for any last minute information that might affect installation choices.
3. Read the `ReleaseNotes.pdf` for an overview of what this release contains that may affect an application that was developed using a prior release of ObjectStore.
4. Read this Installation Guide completely before you perform any installation steps.
5. Ensure that you have Administrator or root priviledges in order to do the installation.

6 Ensure that you have the correct hardware (memory and disk space), operating system, compiler (or C library), JDK (or JRE), application server (if applicable) as described in Chapter 2, System Requirements for ObjectStore, on page 15, and the support information listed in the release notes.

7 Determine the ObjectStore components that will be installed.

8 Determine if the host is to be used for development or not.

9 Determine if the host will run an ObjectStore server process.

10 Determine where the physical databases will reside for development or deployment, and ensure that there is adequate disk space.

11 Determine where the ObjectStore transaction log file will reside which is ideally on a different disk drive than the databases.

If you are planning to use ObjectStore RAWFS databases, ensure that you have raw disk space that is available to be overwritten. If you are at all unsure about whether to use file databases or RAWFS databases, you can discuss the decision with your system administrator or ObjectStore Technical Support. You can defer the creation of an ObjectStore RAWFS partition until later.

12 Determine whether developers will need to modify any existing application source code based upon the information in the Release Notes for this release. (Deprecated APIs or migrating from Release 5.1 of ObjectStore). The upgrade cannot occur until the software has been modified and rebuilt. The application code must be recompiled (but usually not changed in the source code) when migrating from 6.0 to 6.1.

13 Determine whether the databases will need to be either schema evolved or dumped/loaded based upon the type of upgrade of existing database that needs to occur. Developers will need to do this. If the previous release of ObjectStore is 5.1, then a full migration must occur of the application software and the databases; see Upgrading from ObjectStore 5.1 Service Pack x on page 29.

If you are running an earlier release of ObjectStore and are *not* performing a rolling upgrade, shut down any ObjectStore applications and back up your databases, ObjectStore installation binaries and your application software. If you are performing a rolling upgrade, you don't have to shut down any ObjectStore applications but you should back up your databases.

14 Perform the desired type of installation. See Chapter 3, Installing and Configuring ObjectStore, on page 21.

15 Verify the installation.

Getting Help from Technical Support

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

- You have access to the latest revision of the ObjectStore documentation at: http://www.objectstore.net/documentation.
- You can send questions to support@objectstore.net. Remember to include your site ID in the body of the electronic mail message. You can file a support request or question with ObjectStore Technical Support by going to http://www.objectstore.net/support.
- You can call the ObjectStore Technical Support organization to get help resolving problems.

  If you are in North America, call 781.280.4005.

You can access the ObjectStore Technical Support Web site. This site provides access to the following information:

- How to contact ObjectStore Technical Support outside of North America
- A template for submitting a support request. This helps you provide the necessary details
- Frequently asked questions (FAQs) that you can browse and query.
- White papers and short articles about using ObjectStore.
- Sample code and examples.
- The latest version of ObjectStore service packs, and publicly available patches that you can download.
- The Support Matrix on the Technical Support web site (http://www.objectstore.net/support/matrix) lists operating systems and configurations supported by all ObjectStore releases.
- A migration guide for developers who need to migrate from ObjectStore 5.1 to ObjectStore 6.1 at http://www.objectstore.net/documentation/migration.

Chapter 2
System Requirements for ObjectStore

Supported Platforms

The Support Matrix on the Technical Support web site (http://www.objectstore.net/support/matrix) lists the platform configurations that are supported by this release. A supported platform configuration means that it has been thoroughly tested prior to product release. A maintained platform configuration means that it has been known to work and has received some testing, but it has not been tested throughly. See the ObjectStore Technical Support site for more details concerning product support.

Hardware Requirements

CPU

Your CPU must be capable of running the operating system installed. ObjectStore Technical Support recommends using the fastest possible CPU for increased performance.

Memory

The amount of physical memory required will depend on the installation configuration required by the development or deployment environment. The ObjectStore server process does not require much physical memory, but a minimum of 128MB is recommended. If you are running ObjectStore client

processes on a host, you will need more physical memory. Most of the ObjectStore processing is done in the client. If the application is written to require large transactions, then more physical memory will be a benefit. There are some applications that are written with the expected performance of in-memory access speeds of very large datasets. These configurations require a great deal of physical memory.

Disk Space for the Application and ObjectStore

When estimating the amount of disk space required in an ObjectStore environment, consider the following:

- Application binaries and related files — all the files required to run the application (excluding ObjectStore databases and client caches) possibly on a shared device to be used by many clients
- ObjectStore installation files — disk space requirements depend on whether you are installing ObjectStore in a development environment or in a client only environment and, for each environment, what component you install. Also, the amount of disk space occupied by ObjectStore can vary according to the platform on which it is installed.
  - Development environment (can be shared; schema databases (`.adb`, `.ldb`) need to be local to a host running the ObjectStore server process)

ObjectStore DBMS (includes C++ interface)	71MB
ObjectStore Documentation	70MB
Java interface	19MB
Javlin	5MB
DDML	5MB

  - Runtime environment-only (schema databases (`.adb`, `.ldb`) need to be local to a host running the ObjectStore server process)

ObjectStore DBMS (includes C++ interface)	62MB
Java interface	19MB
Javlin	5MB
DDML	5MB

- ObjectStore client cache space — client cache requirements depend on the application

  allocate max. cache size * max. number of clients on host

- ObjectStore transaction log file — maximum size will depend on the application and whether it has large transactions or is using Multi-version Concurrency Control (MVCC). Create this file on a physical disk that is local to the host running the ObjectStore server process and is a different physical disk than all of the databases that will be served by this ObjectStore server process
- Application Databases — make sure that there is enough physical disk space for all databases the application may need (even for future growth). All databases served by this ObjectStore server process must be on the disks that are local to the host where the ObjectStore server process is running. Consider disk space for possible on-line backup, archive logging, replication if the application requires it

Disk Space for Installation Processing

When installing ObjectStore, you also need to allocate disk space for the following:

installation package	180MB
temporary space for the installation (unpackaging)	100MB

The amount of disk space for the above steps, will depend upon the components to be installed. Also, the amount of disk space occupied by ObjectStore can also vary according to the platform on which it is installed.

Note

If you are installing ObjectStore from a CDROM, the installation package does not require additional disk space. However, if you download the installation package from Technical Support, you will need as much as 180MB of additional disk space.

Software Requirements

In order to install this release of ObjectStore, you need to make sure that the appropriate versions of operating systems, C++ compilers/libraries (if C++ client applications will be developed or run), JDKs or JREs, application servers (if Javlin is to be used with an application server), and the appropriate patches for the above software is installed. Information about specific versions and patch-levels for software that supports this release of ObjectStore is documented in the `ReleaseNotes.pdf`.

In addition to the platform configuration information for your installation, it is recommended that you read the product documentation using a Web browser (Netscape 4.7 or greater or Internet Explorer 5.0 or greater) to view the Webhelp or Adobe Acrobat Reader (4.0 or greater) to view the PDFs.

Note

The ObjectStore installation program uses `compress` and `uncompress`, and looks for them in `/usr/bin`. If you are installing ObjectStore on a linux platform, you will need to install `ncompress-4.2.4-31.i386.rpm` to get these utilities.

Installation Media for ObjectStore

Obtaining Installation Media

The files necessary for installing ObjectStore can be obtained either from a CD or from the ObjectStore Technical Support download area. A CD can be obtained by contacting your sales representative or ObjectStore Technical Support.

In order to gain access to the ObjectStore Technical Support download area, you must contact ObjectStore Technical Support (support@objectstore.net or 781 280-4005) to obtain the necessary userid/password.

Each platform configuration for the installation of the ObjectStore components is in its own directory. In addition, on the download site, there will be a directory name which reflects the name of the release of ObjectStore. It is important to understand which platform configuration you need to install.

The installation package contains a number of files, including:

`README.htm`	Important information about this release
`ReleaseNotes.pdf`	New and changed features of this release
`InstallationGuideForUNIX.pdf`	Installation instructions
`osinstal`	Installation program
`*.tar.Z`	Installation contents

The installation package includes additional files, but their contents are self-explanatory.

Description of ObjectStore Releases

ObjectStore releases are given numbers followed by a Service Pack number. For example, ObjectStore 6.1 is the first release and subsequent releases will be named 6.1 Service Pack 1, 6.1 Service Pack 2, etc. Service Packs are *drop-in* compatible and it is expected that customers will upgrade to Service Packs as they are released. In addition to Service Packs, ObjectStore Technical Support sometimes releases *patches* or *quick drops*. These are very specific libraries that are intended to address a software problem that needs to be fixed before a full Service Pack can be tested and released. The installation of Service Packs are full installations or upgrades. The installation of patches or quick drops are typically the replacement of a library. When a Service Pack is released, it contains all patches and quick drops that were distributed prior to the Service Pack.

Chapter 3
Installing and Configuring ObjectStore

This chapter describes how to install ObjectStore and configure the host to use ObjectStore, using the `osinstal` and `osconfig` utilities.

Installing ObjectStore for the First Time

Preparing for Installation

Review the Chapter 1, Overview, on page 1 before you begin the installation. Before you install, determine the following information:

- Location of ObjectStore installation directory
- If there is adequate disk space for installation
- What ObjectStore components need to be installed
- Whether you are doing a development or runtime installation
- Whether this host will be running an ObjectStore server process
- If this host will have an ObjectStore server process:
  - Where the ObjectStore transaction log will reside
  - If the amount of disk space is adequate for the databases
- If you a doing a client-only installation, which host will be used for the ObjectStore server process for the clients and ensure clients will have access to a file system local to that ObjectStore server process.

Installation Procedure

Before you install, make sure that your operating system, compiler, and/or JDK versions are those that are supported for this release. Also make sure that any patches to the operating system, compiler, or JDK have been installed. All information about supported versions and required patches can be obtained from the `ReleaseNotes.pdf` or the Support Matrix on the Technical Support web site (http://www.objectstore.net/support/matrix).

You can install ObjectStore as a non-root user, but this will limit the permissions for which user can use the ObjectStore server to access ObjectStore databases. In addition, the automatic startup of the ObjectStore Cache Manager process cannot be done. In this case the Cache Manager needs to be started manually.

Use the following procedure to install ObjectStore in an environment that is not using Sun Clusters. For information on installing ObjectStore in an environment that uses Sun Clusters, see Installing and Configuring ObjectStore on Sun Clusters on page 39.

1. Change to the top level directory where the installation files are.
2. If you are installing ObjectStore as root, simply enter:

   ```
   ./osinstal
   ```

   If you are installing ObjectStore as a non-root user, enter:

   ```
   ./osinstal -nonroot
   ```

3. You will need to read the License Agreement and acknowledge your acceptance of it. You must type `y` or `yes` if you accept the agreement.
4. You must then enter the path where you wish to install ObjectStore if you want to install ObjectStore in a location other than the default installation directory. The path must end in `ostore`.
5. Once the installation directory is determined or created, you are asked if you want to install a development or runtime environment for ObjectStore. A development environment has all of the runtime executables and libraries, including `ossg`, the utility that enables you to generate C++ schema.
6. You will be asked if you want to install a runtime environment. You should answer `yes` to this question if you want *only* runtime.
7. You will be asked if you want to install a development environment. Answer `yes` to this question if you want to install a development environment which includes runtime.

The installation process now installs the runtime/development environment for ObjectStore DBMS and the C++ development environment.

8 You are now asked if you want to install the Java Interface to ObjectStore. Answer `yes` to this question if your application requires a Java Interface to ObjectStore.

9 You are now asked if you want to install the DDML component of ObjectStore. Answer `yes` if you want DDML installed.

10 You are now asked if you want to install the Javlin component of ObjectStore. Ansyer `yes` if you want Javlin installed.

The installation completes and you are then instructed to set up the ObjectStore environment variables.

You should set up the minimal recommended environment variables before doing any ObjectStore configuration. This includes `OS_ROOTDIR`, `PATH`, and the appropriate library path environment variable (`LD_LIBRARY_PATH`, `SHLIB_PATH`, or `LIBPATH`).

The configuration of ObjectStore is the next step; see Configuration Tasks on page 32. If you have installed the ObjectStore Java component , you must also do some additional configuration steps in order to use it on this host. See Configuring for Java Interface Use on This Host on page 37.

ObjectStore Environment Variables

The following environment variables are required in an ObjectStore environment:

`INCLUDE`	Include path for C++ API: `$OS_ROOTDIR/include` (if C++ API is used) `$OS_ROOTDIR/../osji/include` (if OSJI Java/C++ interoperability is used) `$OS_ROOTDIR/../ddml/include` (if DDML is used)
`LD_LIBRARY_PATH` or `SHLIB_PATH` or `LIBPATH`	Library path for C++ API: `$OS_ROOTDIR/lib` `$OS_ROOTDIR/../osji/lib` (if Java Interface is used) `$OS_ROOTDIR/../ddml/lib` (if DDML is used)
`OS_ROOTDIR`	Installation directory for ObjectStore (ends with `ostore`)
`OS_TMPDIR`	Location of log files used by ObjectStore services

PATH	Path to ObjectStore executables (services and utilities): $OS_ROOTDIR/bin $OS_ROOTDIR/../osji/bin (if Java Interface is used) $OS_ROOTDIR/../ddml/bin (if DDML used) $OS_ROOTDIR/../javlin/bin (if Javlin used)
CLASSPATH	Path to OSJI/JMTL classes: $OS_ROOTDIR/../osji/osji.jar (if Java Interface is used) $OS_ROOTDIR/../osji/tools.jar (if Java Interface is used) $OS_ROOTDIR/../javlin/jmtl.jar (if Javlin is used) $OS_ROOTDIR/../javlin/xerces.jar (if Javlin is used)

Verify Installation

You should verify your installation by confirming that the expected files and directories were created and by running the ObjectStore services or an ObjectStore application, as described in the following sections.Note, however, that before running any ObjectStore service or application, you must perform the configuration tasks described in Configuration Tasks on page 32.

Note

At a later time after you have installed ObjectStore, you can verify that a host was correctly configured and that clients can address the local ObjectStore server running on the host with the following command line:

```
osconfig check -host list_of_hosts
```

For more information, see "osconfig: Configuring ObjectStore" in Chapter 4 of *Managing ObjectStore*.

Verify Installation Directory Contents

Your installation directory contents will depend on what you have selected to install. Some directories will always be installed. You should verify that the following directories or a subset of them have been installed

doc	Documentation in HTML/PDF for all ObjectStore components
ostore	Directory for the ObjectStore DBMS, and C++ environment

ostore/bin	All ObjectStore utilities, including osconfig for performing configuration changes
ostore/etc	Configuration files
ostore/examples	ObjectStore C++ API code examples
ostore/include	ObjectStore C++ API include files
ostore/lib	All ObjectStore libraries, schema databases, and shared libraries
ostore/libsngl	Shared libraries for an ObjectStore Single environment
ostore/unsupported	Objectore performance measurement tools which are available for customers to use, but are not a supported component of ObjectStore
osji	This directory is for the OSJI API of ObjectStore; it contains the OSJI .jar. osji.jar ObjectStore Java interface (OSJI) classes. tools.jar OSJI development tools classes. jdd.jar browser.jar stublib.jar Stubs of OSJI classes that allow user-defined persistence-capable classes to be used in a pure transient manner. For details, see "Using Persistent Classes in a Transient Manner" in Chapter 13 (Miscellaneous Information) in the *Java API User Guide*. The debug versions of the OSJI jar files. They are compiled with the -g flag and have symbolic information that is useful to ObjectStore developers. If you are having a problem, you might be asked to use these jar files to obtain a stack trace. osji_g.jar tools_g.jar jdd_g.jar
osji/lib	Directory of libraries, library schema databases, and shared libraries.
osji/bin	Directory of executables and script files

`osji/include`	Directory of include files needed for C++ interoperability.
`osji/com/odi/demo`	Demonstration hierarchy of examples of ObjectStore programs.
`osji/com/odi/tutorial`	Instructions and sample program for getting started.
`osji/setup`	Script that initializes OSJI after it is installed.
`osji/Setup.class`	Run by the setup script for client-only installations.
`osji/run_person`	Script ot verify the installation by running the `Person` demo.
`javlin`	Installation directory for the Java Middle Tier Library; contains the `.jar` files for using JMTL.
`javlin/bin`	Utilities.
`javlin/com`	Directory of Javlin examples.
`ddml`	Installation directory for the Dynamic Data Modeling Library.

Verify a C++ Development Client

Note

Before verifying a development client, you must perform the tasks described in Configuration Tasks on page 32.

If you have a development environment on this host with a C++ compiler, then you can build and run the `hello2` example by going to `ostore/examples/hello2` and typing:

```
doall
```

Building and running this example verifies that your development environment is set up and you can create an ObjectStore database with the C++ API.

Verify a C++ Runtime Client

Note

Before verifying a runtime client, you must perform the tasks described in Configuration Tasks on page 32.

If you have only a C++ runtime environment, you will not be able to build a C++ application, but simply run one on this host. A simple way that you can verify that you have a C++ runtime client properly installed is to run one of the utilities that come with ObjectStore.

You should make sure that you have set `OS_ROOTDIR`, `PATH` , and the library path environment variables for this to work properly. What you can do to run a C++ client program is to run the `ossize` utility that comes with ObjectStore in the `ostore/bin` directory on an ObjectStore database. Any of the library schema databases that come with ObjectStore can be used for this verification, as in the following command line:

```
ossize $OS_ROOTDIR/lib/metaschm.db
```

The location of the library schema databases depends on which host the ObjectStore server process is running and if this particular ObjectStore server process has been configured for RAWFS database access. If the host for the ObjectStore server process has been configured for RAWFS databases and the library schema databases are in the default RAWFS directory on this host, then you can verify the client on this host (whether a UNIX or Windows host) as follows:

```
ossize <RemoteHostName>::/ostore/schema/6.1/metaschm.db
```

If the host with the ObjectStore server process running has a different location for the library schemas, then you will need to enter that path, rather than the default path shown above.

If the remote host is not configured for RAWFS databases, then you must find out where the schema databases have been installed on that host.

Here is an example of how to access the library schema on the remote host that is UNIX assuming ObjectStore has been installed in `/usr/local` and the schema databases are in the default location:

```
ossize remote_host_name:/usr/local/ODI/ostore/lib/metaschm.db
```

Here is an example of how to access the library schema on the remote host that is Windows assuming ObjectStore has been installed on `C:\` and the schema databases are in the default location:

```
ossize remote_host_name:C:\\ODI\\ostore\\lib\\metaschm.db
```

Verify a Java Client

Note

Before verifying a Java client, you must perform the tasks described in Configuration Tasks on page 32.

To test the installation, enter the following command in the `$OS_ROOTDIR/../osji` installation directory:

```
./run_person
```

This runs the `Person` demo. It displays diagnostic messages if it detects something wrong in the installation.

If you are not running osserver on the machine on which the `$OS_ROOTDIR/../osji` directory is stored, specify a pathname for a database that resides on a machine that is running osserver.

For example:

```
./setup /usr1/bob
./run_person /usr1/bob/person.odb
```

As with the `setup` command, you can specify a host-qualified pathname.

If you are running a UNIX client with a Windows ObjectStore server that requires a user name and password, you can specify them like this:

```
./setup jackhammer:c:\\bob
./run_person jackhammer:c:\\bob\\person.odb bob PassWd
```

Upgrading from ObjectStore 6.0 Service Pack x

It is important that you first read the `README.htm` and the `ReleaseNotes.pdf` to make sure that there are no source code changes that must be made to the application prior to the upgrade. In addition, it is important to ensure that the desired platform configuration is supported for this release.

ObjectStore C++ databases created with 6.0 are compatible with this release provided the same C++ compiler has been used.

Upgrading from 6.0 to 6.1 requires that applications be recompiled and relinked with the 6.1 client libraries.

Before you upgrade, it is necessary that you shut down all ObjectStore applications or services that are running on the targeted host for installation. The cache manager process must also be shut down if it is running. If you want to upgrade your ObjectStore server process without having to shut down ObjectStore clients, see Installing and Configuring for a Rolling Upgrade on page 30.

If the host for installation has an ObjectStore server process and once you have ensured that no applications are accessing the ObjectStore server

process, then you should checkpoint the ObjectStore server process to make sure that all transaction data is propagated from the ObjectStore transaction log to the databases. You can do this as follows:

```
ossvrchkpt host_name
```

You can then shut down the ObjectStore server process.

If this host has an ObjectStore server process, you should back up all databases, application files, and the ObjectStore installation directory.

You should now follow the installation process described in Installing ObjectStore for the First Time on page 21, with the exception of the following. If you currently have an existing ObjectStore installation, you are asked if you want to overwrite the existing installation, rename the existing directory before installing, or delete all files in the existing ObjectStore installation prior to doing the installation. It is recommended that you *not* overwrite existing configuration files in the /etc directory because they may have been modified for your application environment. You might want to save them for reuse with this upgrade of ObjectStore.

You should then do the verification as in described in Verify Installation on page 24.

Lastly, you should verify that your application runs as expected after the upgrade.

Upgrading from ObjectStore 5.1 Service Pack x

Besides the new features and functionality in ObjectStore 6.1, the internal format of the database has changed. What this means is that 5.1 ObjectStore clients and 6.0 or 6.1 ObjectStore clients cannot access the same database. In order for a 6.1 ObjectStore client to access a database created by a 5.1 client, the database needs to be dumped and loaded with the ObjectStore utilities osdump and osload.

This type of upgrade can be summarized with the following steps. You should consult the Migration Guide at http://www.objectstore.net/documentation/migration for detailed instructions on upgrading this release of ObjectStore.

1 Install the 5.1 osdump utility that is compatible with ObjectStore 6.1

2 Install ObjectStore 6.1 in a different directory than 5.1. See Installing ObjectStore for the First Time on page 21.

3 If the application is written in C++, read the Support Note entitled *ObjectStore C++ Migration - Porting from R5.x to R6.x* and migrate your application accordingly

4 Completely rebuild your application (C++ migrated code or Java code) with ObjectStore 6.1 and using an ObjectStore 6.1 server process

5 Back up your 5.1 application and database(s)

6 Dump your ObjectStore 5.1 database(s) using the 5.1 osdump utility from step 1 using the ObjectStore 5.1 server process

7 Load your database(s) using the 6.1 osload utility

8 Test your 6.1 application completely

9 Uninstall ObjectStore 5.1

Installing and Configuring for a Rolling Upgrade

The purpose of a rolling upgrade is to allow for the upgrade of ObjectStore from 6.x to 6.1 or later in a configuration that supports failover. ObjectStore is installed and the ObjectStore configuration process (`osconfig`) is run in a mode whereby the ObjectStore server process is upgraded to the new release first and the clients can continue running. This is possible because the failover configuration allows for ObjectStore server processes to be shut down and restarted with the clients reconnecting as the new ObjectStore server process is brought on-line. In a rolling upgrade scenario, you would upgrade (via `osconfig`) the ObjectStore server processes (each one participating in the failover configuration) and then when it is a convenient time, you would upgrade the clients (via `osconfig`).

Here are the steps for installing and configuring a host running a failover ObjectStore server process with the latest release of ObjectStore:

1 Choose a new `OS_ROOTDIR`, which must be unique in the second-to-last pathname component, because the last pathname component is required to be "`ostore`". For example, if ObjectStore 6.1 GA is currently installed in `/global/opt/ODI/OS6.1.0/ostore`, the new value for `OS_ROOTDIR` could be `/global/opt/ODI/OS6.1.1/ostore`.

2 Run `osinstal` to install ObjectStore software into the new `OS_ROOTDIR`.

The old software and the ObjectStore server process remains running undisturbed during this step.

See Installing ObjectStore for the First Time on page 21 for more information on installing ObjectStore.

3 Run `osconfig` to configure the newly-installed software. This forces the currently running `osserver` to fail over to the newly-installed version. Clients running with the old version of ObjectStore will continue to work. Schema databases in both the old `OS_ROOTDIR` and the new `OS_ROOTDIR` are available through the newly-installed `osserver`. In this situation you should use the `-upgrade` option to `osconfig`. This option causes `osconfig` to ask for the old value of `OS_ROOTDIR` on each machine in the failover pair and to pick up old configuration information from there. Here is an example command line using the `-upgrade` option:

```
$OS_ROOTDIR/bin/osconfig failover -upgrade
```

4 The configuration process will identify the primary host and will ask you to enter the name of the secondary host that is used for failover.

5 The configuration process asks if you want to create symbolic links in `/usr/lib`. In an environment where multiple versions of ObjectStore may be installed, Technical Support recommends that you answer *no* to this question.

6 The configuration process asks if the installation files are going to be used by both ObjectStore server process. In most cases, the answer is *no*.

7 The configuration process asks if the installation directory is accessible over NFS. The answer to this question depends on how this disk is configured.

8 The configuration process then asks if the path for the new installation directory for ObjectStore is the same for both hosts.

9 The configuration process will then ask about locations of the old installations of ObjectStore on each host and if the directories are NFS mounted, etc. Answer all questions concerning the accessibility of the old installations of ObjectStore on each host.

10 The configuration process then asks for confirmation of the information for the upgrading of each of the hosts and then asks for confirmation of the locator file being used.

11 The configuration process then asks if you want the schema files to be used by the ObjectStore server process to be stored in the RAWFS and if so, where in the RAWFS.

12 The configuration process asks where you want the output to be directed for each of the ObjectStore server processes. Each server process has its own output directory, which need not be accessible to other server processes.

13 The configuration process then asks for the names of the directories where file databases will be stored. File databases are not in the RAWFS. The directories (if specified) must be accessible by both ObjectStore server processes. Usually, no directories need to be specified.

14 The configuration process then asks if you want to establish auto startup of the ObjectStore server processes on each host.

15 Existing ObjectStore clients will continue running during this upgrade. If you plan on upgrading the hosts where clients are running then you would perform the following step after all clients on a host have been shut down:

```
$OS_ROOTDIR/bin/osconfig client -host primary_server
```

Client processes can now be restarted one at a time with the new value of `OS_ROOTDIR`, to switch them to the newly-installed ObjectStore libraries. To switch to the new cache manager, the easiest way is to shut down all clients on a machine, shut down the cache manager on that machine, change the `OS_ROOTDIR` value for clients on that machine, and restart the clients, which will launch the new cache manager.

Configuration Tasks

On UNIX, the second required part of the installation is the configuration. Before you begin the configuration, you should make sure that you have installed all required operating system, compiler, and JDK/JRE patches for this release of ObjectStore. These patches are detailed in the Support Matrix on the Technical Support web site (http://www.objectstore.net/support/matrix). The ObjectStore configuration step allows you to set up your environment for client, server or both. Even if you have installed ObjectStore as a non-root user, the configuration of ObjectStore requires root privileges in most cases. If you need to do some configuration as non-root, please contact Technical Support for specific questions about the running of osconfig as a non-root user. There

are some special manual starting of ObjectStore services that will have to occur before you can effectively configure ObjectStore clients.

Setting up a client enables you to create soft links to ObjectStore libraries. Setting up a server enables you to initialize the ObjectStore transaction log file and set up any RAWFS partitions. There are also options for configuring ObjectStore for Sun Clusters 3.0 operating systems, for setting up a failover server and for performing a rolling upgrade of the ObjectStore server process without having to bring down ObjectStore client applications. (The rolling upgrade option is available only when upgrading 6.0 ObjectStore servers to 6.1). You should read about the cluster, failover, and rolling upgrade options in "osconfig: Configuring ObjectStore" in Chapter 4 ("Utilities") of *Managing ObjectStore*.

It should also be noted that configuration is something that can be done at any time. That is if you have done a particular configuration at installation time, it is possible to change the configuration at a later time by running osconfig again. See "osconfig: Configuring ObjectStore" in Chapter 4 ("Utilities") of *Managing ObjectStore*.

If you are installing or configuring ObjectStore in a Sun Clusters 3.0 environment, see Installing and Configuring ObjectStore on Sun Clusters on page 39.

Configuring Host for Client and Server using File Databases (no RAWFS)

This procedure is done after you have installed ObjectStore. It enables this host to run an ObjectStore server process and ObjectStore clients. Even though this host may be configured to run an ObjectStore server process, it need not be set up to run at system startup.

1. Set `OS_ROOTDIR` to the path of the directory where ObjectStore is installed. The directory name ends in `ostore`.
2. If you want the host to use only file databases (no RAWFS), run the following command:

   ```
   $OS_ROOTDIR/bin/osconfig server
   ```

3. You will be asked to confirm the location of ObjectStore that you want to configure.
4. You will then be asked if you want to create soft links in `/usr/lib`. You should respond `no` to this question if you think that you will have an environment where multiple versions of ObjectStore may be installed and

you do not want links in `/usr/lib` to be set to a particular release of ObjectStore.

5 The next part of the configuration will ask you where you want the ObjectStore transaction log file to be created. It is a requirement that an ObjectStore server process have a transaction log file. You should select a path/location for the ObjectStore transaction log file which is on a different disk drive than where the ObjectStore databases will reside (if possible). This is a performance optimization. At this stage of the configuration you should specify the full path for the transaction log file which includes the name of the file. (for example, `/h/devo/ObjectStore/ostore/txn.log`).

6 The next part of the configuration will ask you where you want the schema databases to be physically stored. The default location should suffice. Whatever you chose as the location for the schema databases, they should be on disks that are local to this host.

7 The next part of the configuration is the initialization of the transaction log file. You should say `yes` to the question about initializing the transaction log file.

8 The next part of the configuration is to set up the ObjectStore server process to start up automatically when the system boots and to shut down the ObjectStore server process prior to system shutdown. It is recommended that you set up your configuration to do this automatically, unless you are installing the client and server, but will not be utilizing the ObjectStore server process at this time, but will configure it to be used possibly at a later time.

9 At this point in the configuration, the configuration process will start up the ObjectStore server and the ObjectStore Cache Manager.

At the completion of the server configuration, some automatic verification of the configuration is done and you should get messages that the verification completed and that the ObjectStore Configuration completed.

Configuring Host for Client and Server with RAWFS or File Database capability

This step is done after you have installed ObjectStore. It enables this host to run an ObjectStore server process and ObjectStore clients. Even though this host may be configured to run an ObjectStore server process, it need not be set up to run at system startup. This configuration option allows you to set up a RAWFS for your ObjectStore server process to use.

1. You should run the following if you want this host to utilize file databases or a RAWFS:

   ```
   $OS_ROOTDIR/bin/osconfig rawfs
   ```

2. The configuration process asks you to confirm the location of ObjectStore that you want to configure.

3. The configuration process asks if you want to create soft links in `/usr/lib`. You should respond `no` to this question if you think that you will have an environment where multiple versions of ObjectStore may be installed and you do not want links in `/usr/lib` to be set to a particular release of ObjectStore.

4. It is a requirement that an ObjectStore server process have a transaction log file. The configuration process asks where you want the ObjectStore transaction log file to be created, in the file system or in the RAWFS. You can choose to have the transaction log file in the RAWFS, but your best performance will be if the location of the transaction log file is on a different disk than where the RAWFS partition is.

5. The configuration process asks where you want to locate the RAWFS. The options are platform-dependent. On some platforms, you are given three choices: file, disk, or logical volume.

   You should enter the partition or file path that will be reserved for your RAWFS.

6. If you chose to create a transaction log file in the file system, rather than in the RAWFS, you will be asked for the location of the transaction log file. You should select a path/location for the ObjectStore transaction log file which is on a different disk drive than where the ObjectStore databases (and RAWFS) will reside (if possible). This is a performance optimization. You should specify the full path for the transaction log file which includes the name of the file. (for example, `/h/devo/ObjectStore/ostore/txn.log`).

7. The configuration process asks where you want the schema databases to be physically stored. You can chose to store them in the RAWFS or in a different location in the file system. Whatever you chose as the location for the schema databases, they should be on disks that are local to this host.

8. The next part of the configuration is the initialization of the RAWFS. You should say `yes` to the question about initializing the RAWFS.

9. The next part of the configuration is the initialization of the transaction log file. You should say `yes` to the question about initializing the transaction log file.

10 The next part of the configuration is to set up the ObjectStore server process to start up automatically when the system boots and to shut down the ObjectStore server process prior to system shutdown. It is recommended that you set up your configuration to do this automatically, unless you are installing the client and server, but will not be utilizing the ObjectStore server process at this time, but will configure it to be used possibly at a later time.

11 At this point in the configuration, the configuration process will start up the ObjectStore server and ObjectStore Cache Manager.

12 At the completion of the rawfs configuration, some automatic verification of the configuration is done and you should get messages that the verification completed and that the ObjectStore Configuration completed.

Configuring Host for Client Only

This step is done after you have installed ObjectStore. It enables this host to run only ObjectStore clients.

1 You can configure a client host with either of the following command lines:

```
$OS_ROOTDIR/bin/osconfig client
```

or

```
$OS_ROOTDIR/bin/osconfig client -host list_of_hosts
```

where *list_of_hosts* is a list of one or more space-separated names of hosts that will be running ObjectStore server processes.

2 You are asked to confirm the location of ObjectStore that you want to configure.

3 You are asked if you want to create soft links in `/usr/lib`. You should respond `no` to this question if you think that you will have an environment where multiple versions of ObjectStore may be installed and you do not want links in `/usr/lib` to be set to a particular release of ObjectStore.

4 You are asked if you want to set up the schema databases in a RAWFS on the specified host or hosts.

5 You are asked to specify the location of the schema files on the remote host. The configuration process will then run the `ossetasp` utility on the executables and libraries so that they will point to the schema files.

6 At the completion of the client configuration, some automatic verification of the configuration is done and you should get messages that the verification completed and that the ObjectStore Configuration completed.

Configuring ObjectStore for Failover

If you want to configure ObjectStore for failover, you should:

1 Get two machines with at least one shared disk. This disk will typicaqlly contain a raw-disk Rawfs.

2 Run the `osinstal` utility on both machines.

3 Run the `osconfig` utility.

For detailed information about configuring ObjectStore for failover, see "Failover" in Chapter 6 of *Managing ObjectStore*.

Configuring for Java Interface Use on This Host

1 Determine whether there is an `osserver` process that has access to databases stored in the `$OS_ROOTDIR/../osji` directory. Such an osserver process is either

- Running on the local host
- Running on the host from which the `$OS_ROOTDIR/../osji` directory has been mounted
- Running on some other host that has access to the local host's disks through an ObjectStore locator file

The `osserver` does not have to be available to run the setup program, but it must be available to perform any ObjectStore function.

You can learn whether there is an osserver running for the `$OS_ROOTDIR/../osji` directory by first executing

```
oshostof $OS_ROOTDIR/../osji
```

and then executing

```
ossvrping host_name_returned_by_oshostof
```

2 If no osserver is or will be available to access databases in the `$OS_ROOTDIR/../osji` directory, go to step 3. If an `osserver` is or will be available, go to step 4.

3 If no osserver is or will be available to access databases in the `$OS_ROOTDIR/../osji` directory, run the setup program and specify a directory (or host:/directory) pathname for the location in which you

want to put the OSJI application schema databases. There must be an `osserver` available for this location. After you run the setup program, you must manually copy the OSJI application schema databases to the location you specify. For example, suppose that an `osserver` process is available for databases in the `/usr1/bob` directory. You can enter the `setup` command with a pathname argument like the following:

```
./setup /usr1/bob
```

You can also specify a host-qualified pathname. This is especially useful if the directory is not mounted on the local machine. For example:

```
./setup fuji:/usr1/bob
```

or

```
./setup jackhammer:c:\\bob
```

if `jackhammer` is an NT machine that is running an ObjectStore server.

After you run the setup program, copy the OSJI application schema databases and library schema database to the directory you specified in the setup command. The setup program displays a message telling you to do this. You might need to use commands like `rcp` or `ftp` to copy the databases to a directory that is not mounted on the local machine. If you installed OSJI in the `/opt/ODI/osji` directory, the files you must copy are

```
/opt/ODI/osji/lib/osjcgen.adb
/opt/ODI/osji/lib/libosji.adb
/opt/ODI/osji/lib/libosji.ldb
```

If you have multiple OSJI clients that are using the same remote ObjectStore server, you need to copy these files only once for all clients.

4. Set your `CLASSPATH` environment variable to contain the `osji.jar` and `tools.jar` files, which are contained in the OSJI distribution. Suppose you installed OSJI in the `/opt/ODI/osji` directory. The setting would be

   ```
   CLASSPATH=/opt/ODI/osji/osji.jar:/opt/ODI/osji/tools.jar
   ```

5. Set your `PATH` environment variable to contain the bin directory that is in the OSJI distribution. For example:

   ```
   PATH=/usr/bin:/opt/ODI/ostore/bin:/opt/ODI/osji/bin
   ```

6. Set your library path environment variable to contain the `lib` directory that is in the OSJI distribution. For example:

   ```
   LD_LIBRARY_PATH=/opt/ODI/ostore/lib:/opt/ODI/osji/lib
   ```

7. Ensure that an `osserver` is available for all directories in which you want to store databases.

Installing and Configuring ObjectStore on Sun Clusters

1. Run `osinstall -cluster` to install ObjectStore software in a Sun Clusters environment. If installing into the global file system as recommended, this step only has to be done once and can be done on any node in the cluster.
2. Make sure you have the tar file `EXLNossvr.tar`, which contains the Sun Clusters agent that monitors and controls the `osserver` process. The tar file can be found in the `ostore` directory of the installation package. You will have to untar `EXLNossvr.tar` before you can use it.
3. Run `pkgadd` to install the `EXLNossvr` package. Sun requires this step to be repeated on each node in the cluster. This can be done either before or after step 1.

   ```
   pkgadd -d EXLNossvr
   ```

4. Run `osconfig` with special arguments that tell it to configure for a cluster.

   ```
   $OS_ROOTDIR/bin/osconfig cluster -logical_host list_of_hosts
   ```

   The host name is a logical host name dedicated to use by this `osserver` resource, not the host name of the node in the cluster. (See *Sun Cluster Data Services Installation and Configuration Guide* from Sun.)

   You specify two logical host names if you want to configure the failover system to consist of two `osserver` processes, each running on a separate machine.

5. You will be asked if you want to use the default Sun Cluster utilities, if not then you must specify a path to the Sun Cluster utilities that you will be using.
6. You will be asked if you are using any local filesystems that reside on the cluster-global disk groups. If you are, you will be asked if they are mounted on this cluster node. If you have not mounted them, they must be mounted.
7. A search will be done for existing cluster-nodes which will be listed. You will be asked to rank them.
8. You will then be asked if you want to configure your environment for RAWFS. If you have reserved a device for RAWFS use, you should specify a global device path that will be accessible by all nodes of the cluster.
9. It is a requirement that an ObjectStore server process have a transaction log file. The next part of the configuration will ask you where you want

the ObjectStore transaction log file to be created, in the file system or in the RAWFS You can choose to have the transaction log file in the RAWFS, but your best performance will be if the location of the transaction log file is on a different disk than where the RAWFS partition is. If you chose to have the transaction log file in the file system, it must be in a directory that is on a shared disk that is accessible to all nodes of the cluster.

10 You will next be given the option of creating a file in a directory in your file system (choice 1) to contain the RAWFS or specifying the raw disk partition (choice 2) that will be reserved for your RAWFS. You should enter the partition or file path which will be reserved for your RAWFS.

11 If you chose to create a transaction log file in the file system, rather than in the RAWFS, you will be asked for the location of the transaction log file. You should select a path/location for the ObjectStore transaction log file which is on a different disk drive than where the ObjectStore databases (and RAWFS) will reside (if possible). This is a performance optimization. You should specify the full path for the transaction log file which includes the name of the file. (for example, `/h/devo/ObjectStore/ostore/txn.log`).

12 The next part of the configuration will ask you where you want the schema files to be physically stored. You can chose to store them in the RAWFS or in a different location in the file system. Whatever you chose as the location for the schema databases, they should be on disks that are accessible to all nodes of the cluster.

13 You will be asked where you want the output of the ObjectStore server process to be written. This should be a directory that is accessible to all nodes of the cluster.

14 You will be asked which directories will be used for all of the ObjectStore databases to be created and accessed by your applications. These directories must be on one or more disks that are accessible to all nodes of the cluster.

15 At this point the configuration process lists all of the configuration choices you have made and asks for confirmation. Enter yes if you want to continue with the configuration.

16 The configuration process now initializes the RAWFS or upgrades it if necessary and asks you for confirmation before doing so.

17 The configuration process checks for existence of the resource group and re-registers it. You are asked to confirm both of these.

18 Once the resource group is registered, the ObjectStore server process is started up and the necessary schema is copied from the installation

directory to where you requested it be located earlier in this configuration dialog.

19 You will then be asked if you wish to test each node in the cluster. ObjectStore Technical Support recommends that you do this test.

20 Configure and install clients with the following command line:

```
$OS_ROOTDIR/bin/osconfig client -host list_of_hosts
```

where *list_of_hosts* is a list of one or more space-separated names of hosts that will be running ObjectStore server processes. The names should be the same as those used earlier.

21 If the clients run on the cluster and are "highly available," they have their own resource groups, which should be set up to declare a dependency on the osserver resource group(s) so `osserver` will start first.

22 Copy the locator file generated by `osconfig` to the client machines, possibly merging it with other locator file information desired at the site. ObjectStore must be installed on the client machines first.

Documentation specific to Sun Cluster commands can be found in the `docs` directory where the `EXLNossvr` package is installed, for example, `/opt/EXLNossvr/docs`.

Using a Rolling Upgrade to Install and Configure ObjectStore on Sun Clusters

The version of ObjectStore installed on a cluster can be upgraded without turning off service for longer than the time required to failover, by following this procedure:

1 Run `osinstall -cluster` to install ObjectStore software in a Sun Clusters environment. If installing into the global file system as recommended, this step only has to be done once and can be done on any node in the cluster.

2 Make sure you have the `EXLNossvr` package, which is the Sun Clusters agent that monitors and controls the `osserver` process.

3 Run `pkgadd` to install the `EXLNossvr` package. (The package `EXLNossvr.tar` will need to be untarred from the installation files on the CD or the download.) Sun requires this step to be repeated on each node in the cluster. This can be done either before or after step 1.

```
pkgadd -d EXLNossvr
```

4 Run `osconfig` with special arguments that tell it to configure for a cluster.

```
$OS_ROOTDIR/bin/osconfig cluster -logical_host host_names
  -upgrade
```

The host name is a logical host name dedicated to use by this `osserver` resource, not the host name of the node in the cluster. (See *Sun Cluster Data Services Installation and Configuration Guide* from Sun.)

5 You will be asked if you want to use the default Sun Cluster utilities, if not then you must specify a path to the Sun Cluster utilities that you will be using.

6 You will be asked if you are using any local filesystems that reside on the cluster-global disk groups. If you are, you will be asked if they are mounted on this cluster node. If you have not mounted them, they must be mounted.

7 A search will be done for existing cluster-nodes which will be listed. You will be asked to rank them.

8 You will be asked for the old `OS_ROOTDIR` that is currently being used by the ObjectStore server process.

9 The next part of the configuration will ask you where you want the schema files to be physically stored. You can chose to store them in the RAWFS or in a different location in the file system. Whatever you chose as the location for the schema databases, they should be on disks that are accessible to all nodes of the cluster.

10 You will be asked which directories will be used for all of the ObjectStore databases to be created and accessed by your applications. These directories must be on one or more disks that are accessible to all nodes of the cluster.

11 The configuration process modifies the resource properties and forces the ObjectStore server process to restart with the new properties. Then it copies schema databases from the installation directory to the location you specified earlier, modifies executables to point to the schema databases in this location, and verifies the schema databases.

12 The configuration process checks for existence of the resource group and re-registers it.

Once the resource group is registered, the ObjectStore Server process is started up and the necessary schema is copied from the installation directory to where you requested it be located earlier in this configuration dialog.

13 You will then be asked if you wish to test each node in the cluster. ObjectStore Technical Support recommends that you do this test.

14 When you want to upgrade the clients, configure and install the clients with the following command line:

```
$OS_ROOTDIR/bin/osconfig client -host list_of_hosts
```

where *list_of_hosts* is a list of one or more space-separated names of hosts that will be running ObjectStore server processes. The names should be the same as those used earlier.

15 If the clients run on the cluster and are "highly available," they have their own resource groups, which should be set up to declare a dependency on the `osserver` resource group(s) so osserver will start first.

16 Copy the locator file generated by `osconfig` to the client machines, possibly merging it with other locator file information desired at the site. ObjectStore must be installed on the client machines first.

Documentation specific to Sun Cluster commands can be found in the `docs` directory where the `EXLNossvr` package is installed, for example, `/opt/EXLNossvr/docs`.

Troubleshooting Installation Problems

If you run into a problem during installation or configuration of ObjectStore, contact Technical Support; see Getting Help from Technical Support on page 13. There are a number of ObjectStore FAQs available to users that might help with particular installation problems so it is recommended that you check the FAQs.

Chapter 4
Miscellaneous Information

Uninstalling ObjectStore

In order to uninstall ObjectStore, you must make sure that any databases (whether in the file system or in a RAWFS) are backed up in the event that you may want any of your data in the future.

Uninstalling ObjectStore involves removing all files installed with ObjectStore. Before removing these files, you should make sure that all ObjectStore applications are no longer running that might be using the ObjectStore server process on this host. Also, you should make sure to shut down any ObjectStore Cache Manager or ObjectStore server processes that are running, as follows:

```
cd $OS_ROOTDIR/bin
oscmshtd
ossvrshtd <hostname>
```

Once these processes are no longer running, you can remove the contents of `.$OS_ROOTDIR/..` and its descendants.

Finally, you should remove the file in the `/etc/rc2.d` directory which refers to the `startup/shutdown` of ObjectStore. This file has `ostore` in its name.

Adding a Component

The installation process allows you to install these ObjectStore components:

- OSJI
- DDML
- Javlin

If you do not install these components during the installation process, you can install them at a later time.

Before installing any of these components, you must ensure that `OS_ROOTDIR` is properly set.

To install OSJI, run the following command lines from the installation package:

```
cd osji
osjinst
```

To install DDML, run the following command lines from the installation package:

```
cd ddml
ddmlinst
```

To install Javlin, you must first install OSJI and then run the following command lines from the installation package:

```
cd javlin
javlininst
```

Resource Group Properties for Sun Clusters

The following changeable resource group properties are of interest in an osserver resource group. All of these are standard.

Nodelist : stringarray

Default is all cluster nodes in arbitrary order

These are the nodes where this resource group can run. Nodes earlier in the list are more preferred than nodes later in the list. It is important to set this when failover is configured as two `osserver` processes, and to set it to

different values in the two resource groups, so that in the absence of node failures the two osserver processes will run on different nodes.

Failback : Boolean

Default is `True`

If true, when a node comes online that is more preferred than the node where the resource group is currently running, the resource group will be moved to the more preferred node. This will look like a failover to clients.

Pingpong_interval : integer (seconds)

Default is `3600` (one hour)

Approximately: If a failure has occurred on a node more recently than this, avoid using that node. For the exact meaning, see the Sun documentation.

RG_description : string

Default is `ObjectStore server for logical host <name>`

Human-readable identification.

Resource Properties for Sun Clusters

The following changeable standard resource properties are of interest in an osserver resource.

Thorough_Probe_Interval : integer (seconds)

Default is `60`

Interval between ossvrping's.

Probe_timeout : integer (seconds)

Default is `30`

Time out value for the probe (seconds)

FailOver_Mode : enum { NONE, SOFT, HARD }

Default is `SOFT`

Controls what happens if starting or stopping osserver fails, see the Sun documentation.

Network_resources_used : stringarray

Default is "" (but the register_osserver script sets this to the correct value)

Port_list : stringarray

Default is <empty>

Start_timeout : integer (seconds)

Default is 300

Stop_timeout : integer (seconds)

Default is 60

Validate_timeout : integer (seconds)

Default is 300

Update_timeout : integer (seconds)

Default is 300

Monitor_Start_timeout : integer (seconds)

Default is 60

Monitor_Stop_timeout : integer (seconds)

Default is 60

Monitor_Check_timeout : integer (seconds)

Default is 300

The timeout for invoking the validation method that is called before doing a monitor-requested failover.

Retry_Count : integer

Default is 5

The number of times a monitor attempts to restart the resource if it fails.

Retry_Interval : integer (seconds)

Default is 3600

The number of seconds over which to count attempts to restart a failed resource.

Monitor_retry_count : integer

Default is `4`

Number of PMF restarts allowed for the fault monitor

Monitor_retry_interval : integer (minutes)

Default is 2

Time window (minutes) for fault monitor restarts

R_description : string

Default is `ObjectStore server for logical host <name>`

Human-readable identification. The system default is `""` but we will default it to something useful in the script that creates the resource(s).

ObjectStore-Specific Resource Properties

The following changeable ObjectStore-specific resource properties are of interest in an osserver resource.

OS_ROOTDIR : string

Default is `/global/opt/ODI/OS6.1.0/ostore`

This is where ObjectStore was installed. If it is not a global file system path, it needs to be a node-local path in which ObjectStore has been installed on every node on Nodelist. The value of `OS_ROOTDIR` must be the same on every node.

osserver_command : string

Default is `$OS_ROOTDIR/lib/osserver -hostname $HOSTNAME -p file`

This is the command that is executed to start the `osserver` process. It cannot contain I/O redirection, $ substitution of variables other than `OS_ROOTDIR` and `HOSTNAME`, or other complex syntax.

`$OS_ROOTDIR` is substituted from the `OS_ROOTDIR` resource property.

`$HOSTNAME` is substituted from the logical host name that was specified with `register_osserver -h`.

The -p argument specifies a location in the global file system for the server parameter file. This ensures that the same server parameter file is used no matter which node is currently running the osserver process.

Environment : stringarray

Default is <empty>

A stringarray of var=val environment settings for osserver. The monitor process also sees these environment settings. Use this for any customization you need to do.

You do not need to put OS_ROOTDIR, PATH, and LD_LIBRARY_PATH in here.

Location of Schema Databases

If you decide to put the schema databases in a different location after configuring ObjectStore, use the ossetasp utility to update the following executables and libraries to point to the correct schema databases:

- libosalloc
- liboscmp
- liboscol
- libosqry
- libosse

Use the following command line to display the pathname of the current location of an application schema database (.adb) file:

```
ossetasp -p executable_or_library
```

Use the following command line to assign a new location to be used by any of the executables or libraries listed above:

```
ossetasp executable_or_library new_adb_location
```