OServ (ObjectServer) 2.1
User's Manual

2012/08/02

uewxiun@pb3.so-net.ne.jp
Takahiko Tanaka

OServ (Object Server)

This manual was written for OServ(Object Server) 2.1. This manual assumes that all sample code is understood by reader that has basic knowledge of Java language and programming skill. This describes methods such as installation, execution, development and so on. OServ (Object Server) 2.1 is one of the most powerful software for distribution of user's mechanism. It provides features to hold, execute, save and restore static data. It supports for security feature to restrict access of file. OServ(Object Server) 2.1 supports Serializable object as parameter of distributed method. And it also supports some Input/Output streams involving contents. It doesn’t let be aware of OServ(Object Server) 2.1.

2012/06/11 Takahiko Tanaka

Software license

 You can pay fee for purchases of software. Package includes source programs. You can see, modify or embedded them into other program. But you must not use for malicious purpose or for preventing from updating.

Representative of this software

If you have any question, please let me know via e-mail. The address of e-mail is as like follows,

uewxiun@pb3.so-net.ne.jp

Environment

Java 1.5.0_13
OS - Mac OS Darwin 9.6.1
Unix, Linux, Solaris, Windows

1. ObjectServer Feature

OServ(Object Server) is the most powerful server software for managing customer's server software. It encourages making software systems based on server-client model in Java language. This system provides a feature that generates stub class to connect to server. User only makes interface class in Java language and execute the generator.

A stub class the generator generates is used as server class. User doesn't need to change user source code to switch from real server class to stub class. So user runs system without consciousness on network. And this system also provides a persistent feature. So user doesn't take care of daemon software for holding the newest data. This server provides both the execution service and the persistent service.

2. Installation

OServ.jar.zip archive file is distributed. OServ.jar.zip file is a archive in zip format. User must deploy by executing unzip command.

```
# cd WORKING_DIRECTORY
# unzip OServ.jar.zip
```

To execute ObjectServer server software, you must specify OServ.jar archive file in classpath option.

Server side instruction

```
# mkdir SERVER
# cd SERVER
# cp ../OServ.jar .
# java -classpath ./OServ.jar oserv.ObjectServer
```

note: User must set write permission to directory that ObjectServer runs.
Because it makes a directory that is called persistent.
It will hold user server code that user will register.

Test and Example

Following step is needed for execution of client samples.

```
# mkdir CLIENT
# cd CLIENT
# cp ../OServ.jar .
```

How to register user's server code.

```
# mv ../SampleServer.jar .
# mv ../PersistentServer.jar .
```

```
# java -classpath ./OServ.jar oserv.ObjectTool sample.SampleServer
# java -classpath ./OServ.jar oserv.ObjectTool test.PersistentServer
```

(java –Duser.home=`pwd`/sample/ -classpath ./OServ.jar oserv.ObjectTool sample.SampleServer
When SampleServer.jar is located under sample directory in current directory.)

```
# java -classpath ./OServ.jar oserv.ObjectTool -l
```

(java -classpath ./OServ.jar -Doserv.service.hostname=localhost oserv.ObjectTool -l)

(When you want to remove registered package, you can specify -d option with package's name as like follows,
java -classpath ./OServ.jar oserv.ObjectTool -d sample.SampleServer)
If you want to update the package, you can overwritten executing normal registation.

How to execute user's client code.

mv ../userclient .
cd userclient
java -classpath ../OServ.jar:./ sample.SampleClient
(When the command execute on remote host, its syntax is as like follows,
java -Dserv.service.hostname=server_hostname -classpath ../ OServ.jar:./ sample.SampleClient)
java -classpath ../OServ.jar:./ test.PersistentClient

How to generate stub in user's client side.

rm test/PersistentServer.java test/PersistentServer.class
java -classpath ../OServ.jar:./ oserv.ObjectGenerator PersistentService test.PersistentServer
javac -classpath ../OServ.jar:./ test/PersistentServer.java
java -classpath ../OServ.jar:./ test.PersistentClient

How to verify data condition via network transfer.

cd ../../CLIENT
mv ../var .
java –Duser.home=`pwd`/var/server –classpath ./OServ.jar oserv.ObjectTool var.VarServer
java –classpath ./OServ.jar:./var/client var.VarClient all
(all means to execute all items. Each item means as like follows,

Receiving tests for native variable	1-8
Receiving test for VarException	9
Receiving test for String	10
Receiving tests for object variable	11-18
Receiving test for SerializableObject	19
Receiving test for Object	20
Sending tests for native variable	21-28
Sending test for Object	29
Sending test for String	30
Sending tests for object variable	31-38
Sending test for SerializableObject	39
Receiving test for byte array	41
Sending test for byte array	42

)

Correct value.

Variable test or String test results in number same as test number. Only test for boolean variable is true. VarException test displays Exception message. Test for SerializableObject displays name of object. Test for byte array get contents of array that is initialized as index number.

Incorrect value

Tests from 1 to 8 display -1 value. Other tests display null or exception message.

3. Command Reference

This section is written for commands that are involved in OServ package. Command's usage and summary of feature are collected. OServ was written in Java. So all of commands is executed under java vm as like follows,

```
java –classpath archive.jar:directory module_class
```

Command syntax corresponds to module_class part that involves arguments. And module_class is compiled class file of Java.

ObjectServer

Syntax

```
oserv.ObjectServer
```

Summary

ObjectServer watches over two specific ports running as daemon process. One of them is used for registration request. The other of them is used for execution request. Both requests are executed in independent thread of Java. ObjectServer supports for a feature to hold persistent data's value. ObjectServer is one of server program that manages execution of a method in a class. Persistent function is implemented by saving or restoring static data of the class. And security feature is available for each registered service class. The method in the class must use user.dir property value via getAbsolutePath() method in File class. Normally, when File class is created with relative path, it operates under current directory even if return value from getAbsolutePath() method contains value of user.dir. So developer must call constructor of File or other file operation class such as FileOutputStream with the absolute path again.

ObjectTool

Syntax
oserv.ObjectTool [-d class | -l | -L | class]

Summary
ObjectTool issues a registration request. It also has features for reference to registered services and for deletion of service. It always needs parameter to select behavior. User must make a archived file in jar. It must include two files that are compiled service class and compiled interface class.

class
This class name should be include package name in java class hierarchy. Archived file formatted in Jar should be located under execution directory. Archived file should be named after package name of class or class name.

-d class
ObjectTool issues a request to delete registered class from ObjectServer.

-l
ObjectTool issues a reference request to ObjectServer and it displays list of registered classes.

-L
ObjectTool issues a reference request for forwarding information to ObjectServer and it displays list of forwarding information.

ObjectSystem

Syntax
  oserv.ObjectSystem [-p | -s] class

Summary
  ObjectSystem command provides a feature for switching between parallel execution mode and serial execution mode. The serial execution mode of service means to refuse a multiple request. System executes only one request at a time in serial execution mode.

-p
  Parallel execution mode.

-s
  Serial execution mode.

class
  class is name of registered class.

ObjectKill

Syntax
oserv.ObjectKill [-l | -s | ids]

Summary
ObjectKill issues a system service request. It also has features for reference to threads that serve user service requests. When it doesn't have any arguments, it display list of thread name that involves id.

-l
execute Infinite loop for test. This command creates a service thread that is referred to by ObjectKill command.

-s
This option should be used for referring system threads.

Ids
Id means Thread number in Java VM. It is normally referred as Thread name.
When ids are specified, ObjectKill issues termination requests to ObjectServer.

ObjectService

Syntax
oserv.ObjectService [-f target file] service

Summary
It gets a service archive file or a target file in service archive file from remote server via forward service.
User can get interface file without information of the service host.

-f target file
Target file should be involved in the archive file that is specified by service name.

service
This argument specifies a name of registered service.

ObjectGenerator

Syntax

oserv.ObjectGenerator interface class

Summary

It generates stub class file that it connects to. Stub class file should be implemented based of interface specified. ObjectGenerator is executed for creating stub class. The created stub class involves interface class specifying with package hierarchy. So user can execute a program with the stub class instead of the real service class. The stub class calls ObjectClient class internally. And it will connect to ObjectServer for executing the real service class.

It takes interface class and stub class. If the real server class has a package hierarchy, the stub class name should be specified hierarchy with the package.

it makes a package directory and copy an interface class into it when it is executed on directory that has the class file of interface and different name from name of package.

When OServ runs, ObjectGenerator can extract interface file from OServ. So user doesn't need to locate interface file before executing ObjectGenerator command for creating source file of stub class.

interface

Stub class must include an interface for specifying abstract function.
(It allows to specify class name with '/' separator when class name doesn't include it.)

class

Class name of service that involves package hierarchy should be specified.

4. Architecture

4.1 Summary

OServ(ObjectServer) softeare plays a role for network communication instead of client procedure. It is difficult to implement network procedure for network communication so that it is hard and heavy.

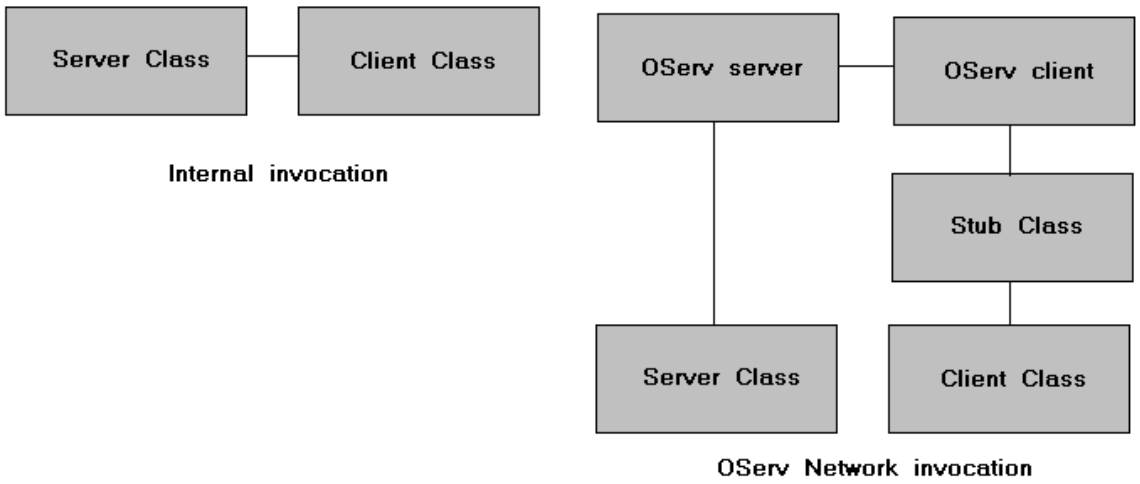

Figure 1 Relation between classes.

In figure 1, OServ server means ObjectServer class. OServ client is implemented as ObjectClient. And ObjectTool performes to register in order to connect between ObjectServer class and Server class. ObjectGenerator makes Stub class file.

4.2 Forward feature

Forward feature is one of features OServ provides and is useful for concealing real servers. In this manual, this feature involves exchanging information of services. User can configure server only to set up server list without any service.

Concept
As CPU (Central Processing Unit)’s performance was very poor and machine's reliability was very low a coupe of decade ago, so many application programs have features for distributing their work. It was important to conceal location of execution

and to dispatch to others. Nowadays, there are a lot of machines whose performance is very strong. But user needs to gather services to administrate efficiently. And user needs to provide a service that is persistent. Therefore object server provides this feature.

Mechanism
OServ (Object Server) provides simple client-server relationship for service. Forward feature expands it to three layers service that is client-forward-server. And forward feature delegates of receipt for services. In short word, only one forward hosts can receive from and respond to all of clients instead of all of servers. And server allows to inform clients to receive new service. User doesn't need to set up or restart for updating information.

Tunnel feature
OServ provides a feature to forward a request. It is useful for machines in separated network.

Client Feature
OServ asks list of service to server when oserv.forward.servers is configured. Forward client serves a remote service to own user directly.
The format of oserv.forward.servers is as like follows,

oserv.forward.servers=oserv://host name or ip address:port number/, oserv://host name or ip address:port number/

In the format, an element of host is formatted as URI. The scheme name is OSERV. Port number should be adopted value configured as oserv.request.port in a server. The element is concatenated with comma separator.

Server Feature
OServ posts their own registered service when oserv.forward.clients is configured.

Security Feature
OServ provides a feature to reject a request for connection.
It allows user to configure as like follows,

oserv.forward.refuse=oserv://172.23.43.10/,oserv://164.44.0.0/?netmask=0xffff0000

When forward client or forward server accepts request from other machine, it refers to the table configured. When source host is registered, it rejects the request. User can also configure network address with network mask value in the query option field.

5. How to implement

5.1 Implementation method

Distribution
For the first of all, it is focused on to separate a procedure. Service procedure in server side is implemented as common and staple. Client procedure is a role as user's interface and interpreter of user's request.

Definition
Next process is to define an interface for communicating between service procedure and client procedure. Java language environment supports abstract feature for controlling implementation. Interface definition means to force program that include it to implement functions that interface lists with arguments. Implementation is as like follows,

```
public interface PersistentServer {
	public String counter();
}
```

Compiler program can detect failure of unmatched call and display messages based on the failure.

Implementation
Service class must include interface class after implements clue. Compiler program will issue an error message when abstract function interface class lists is unmatched. Serializable Object is allowed to use as parameter or return value.

```
public class PersistentServer extends Object implements PersistentService {
	public String counter() { }
}
```

Implement OServ Method
OServ (ObjectServer) recognizes OServ method as special handler for initialization. OServ execute it when OServ receives a registration request and OServ restarts.

Syntax of OServ is as like follows,

public void OServ();

It will be implemented as like follows,

```
static Boolean initial_flag;
public void OServ() {
      initial_flag = true;
}
```

OServ doesn't take any exception OServ() method generate.

5.2 Migrate to distribution mechanism

User must replace service class with stub class having a feature that change invocation procedure to communication procedure between it and client class. User doesn't modify code of client class when one makes stub class named same as service class. ObjectGenerator command allows user to do it. ObjectGenerator command takes two arguments. One of them is name of interface class. The other of them is name of server class.

For example, stub of PersistentServer sample class is made as like follows,

```
# java -classpath ./OServ.jar oserv.ObjectGenerator PresistentService
PersistentServer
```

First argument is an interface named as PersistentService. Second argument is stub name for replacing server with. In implementation of client (PersistentClient.java), server class is
used as like follows,

>>PersistentClient.java

```
package test;

import java.lang.*;
import java.io.*;

public class PersistentClient extends Object {
	PersistentServer pserv;
	public void PrintCounter() {
		String vl;
		pserv = new PersistentServer();
		vl = pserv.counter();
		System.out.println("Persistent Counter " + vl);
	}
	public static void main(String[] arg) {
		PersistentClient pcl = new PersistentClient();
		pcl.PrintCounter();
	}
}
```

When variable that is held all over requests, it must be declared as a static variable. Real implementation is as like follows,

>>PersistentServer.java

```
package test;

import java.lang.*;
import java.io.*;

import test.*;

public class PersistentServer extends Object implements PersistentService {
	public static Integer cvalue = new Integer(0);
	public String counter() {
		int ct = cvalue.intValue();
		ct = ct + 1;
		cvalue = new Integer(ct);
		return String.valueOf(ct);
	}
}
```

ObjectGenerator needs compiled interface class for making the stub class.

5.3 Stream Parameter

Stream object is available as parameter or return value in ObjectServer. This feature is an artificial feature. When ObjectServer detects input stream parameters, ObjectServer copies data into files it creates. Invoked method receives the input stream objects ObjectServer creates. If output stream parameters are specified as parameter, ObjectServer creates output stream objects and transfers them to method ObjectServer invokes. After executing the method, ObjectServer sends back contents of output stream objects to the client side and writes them into output stream objects client program specifies. ObjectServer supports following input/output stream java supports.

java.io.InputStream
java.io.FileInputStream
java.io.DataInputStream
java.io.BufferInputStream
java.io.ByteArrayInputStream
java.io.PushbackInputStream
java.io.SequenceInputStream
java.io.LineNumberInputStream
java.io.FilterInputStream

java.io.OutputStream
java.io.FileOutputStream
java.io.DataOutputStream
java.io.BufferOutputStream
java.io.ByteArrayOutputStream
java.io.FilterOutputStream
java.io.PrintStream
java.io.PipedOutputStream

5.4 Transient Feature

Transient feature provides and execution environment and common working directory. It is the best way to manipulate data in the common working directory. User executes function same way as registered service even if user doesn’t need to registration of server module. User just specifies location of service archive file as a property. Following example is a case where service archive is located in current working directory. User needs stub class for a request in the case. In the following case the stub is made under sample directory.

```
# java –Doserv.transient.home=`pwd` -classpath ../OServ.jar:./
  sample.SampleClient

# ls
SampleServer.jar sample
```

5.5 Security Feature for operation of file

OServ (Object Server) restricts operation of file using security manager in Java language. Service class that user implements can only operate files under a directory OServ allow to use. Service class can know it using user.dir property. java.lang.File class that Java provides normally provides translation mechanism. Service class must translate from relative path to absolute path using getAbsolutePath() method. The real code is as like follows,

```
File fd = new File("");
String user_dir = fd.getAbsolutePath();
fd = new File(user_dir + file.separator + file_path);
```

or

```
File fd = new File(file_path);
String absolute_path = fd.getAbsolutePath();
fd = new File(absolute_path);
```

Access violation happens when service class operate file for reading, writing or deleting.

6. Configuration

OServ (Object Server) is based on TCP/IP socket and it provides a service to execute object dynamically.

```
oserv.table.size=100
oserv.debug.level=5
oserv.request.port=2000
oserv.connection.timeout=10000
oserv.retry.count=10000
oserv.max.thread=40
oserv.buffer.maximum=1000
oserv.service.hostname=localhost
oserv.log.file=tty
oserv.user.work=userwork
oserv.forward.servers=oserv://localhost:2001/
oserv.forward.clients=oserv://localhost:2002/
oserv.forward.tunnel=1
oserv.forward.refuse=oserv://other/
oserv.request.refuse=oserv://other/
oserv.object.refuse=oserv://other/
oserv.system.forbidden=0
oserv.message.language=jpn
```

oserv.table.size is a property to specify size of oserv table. It is also used as extansion unit.

oserv.debug.level is a property to specify which log message it takes. 3 means to take normal log message and 5 means to take messages for debug.

oserv.request.port is a property to specify port number of socket.

oserv.connection.timeout is a property to specify time to expire. It will set as option of socket.

oserv.retry.count is a property to specify maximum count of retry for receiving data via network.

oserv.max.thread is a property to specify maximum number of thread. OServ allocates the thread table based on it. It ranges from 10 to 1000.

oserv.buffer.maximum is a property to specify size of buffer area. It will be used

when object content is transfered.

oserv.service.hostname is a property to specify name of host.

oserv.log.file is a property to specify the place which it puts log messages to.

oserv.user.work is a property to specify the working directory for service. Relative expression means to locate the directory under the directory oserv.work.path specifies. Default value is userwork. If there isn't the directory, OServ create it automatically. Absolute path is used directly.

oserv.forward.tunnel is a property to restrict number of layer. When user can configure a client for a server that has only forward services, user restricts number of layer using this property. In other words, user prohibits a client for forward server by specifying oserv.forward.t oserv.forward.tunnel is a property to restrict number of layer. When user can configure a client for a server that has only forward services, user restricts number of layer using this property. In other words, user prohibits a client for forward server by specifying oserv.forward.tunnel=1. oserv.forward.servers is a property to specify server list of object service. The object server that is configured with this key asks servers if they have some service. And the object server sets up remote service table for forwarding a service that it receives.

Oserv.forward.servers is a property to specify a list of servers which serve services. When the object server runs, it gets information from these servers. The servers in list should be separated by comma and each entry is URI expression.

oserv.forward.clients is a property to specify a list of clients which object server serves. When the object server registers a new service, it broadcasts its information to all clients based on the list. The machines in list should be separated by comma and each entry is URI expression.

oserv.request.refuse is a property to specify machines or networks to deny request access. The machines or networks in list should be separated by comma and each entry is URI expression. URI expression is used for specifying network too. And URI of network must involves netmask key as query string as like ‘?netmask=0xffff0000’. The number of netmask indicates valid bit in network address.

oserv.object.refuse is a property to specify machines or networks to deny registration access. The machines or networks in list should be separated by comma and each entry is URI expression. URI expression is used for specifying network too. And URI of network must involves netmask key as query string as like ‘?netmask=0xffff0000’. The number of netmask indicates valid bit in network

address.

oserv.forward.refuse is a property to specify machines or networks to deny forwarding access. The machines or networks in list should be separated by comma and each entry is URI expression. URI expression is used for specifying network too. And network must involve netmask key as query string as like ‘?netmask=0xffff0000’. The number of netmask indicates valid bit in network address.

oserv.system.forbidden is a property to specify system condition to allow to use system services. The system services are used for referring and terminating thread in ObjectServer. It should be specified as 1 for prohibiting. 0 means to allow to use. The default value is 1.

Oserv.message.language is a property to specify language of system messages. It accepts jpn and eng only. The jpn means to specify Japanese language. The eng means to specify English language.

Above properties are defined in sample oserv/oserv.properties.
oserv.work.path is new property to specify location of working directory.

Appendix A. Counter Example

This chapter describes how to develop a service that is registered into OServ (Object Server). This describes about mechanism of a procedure of property on client program. OServ (Object Server) allows service to have specific oserv.properties file for controlling their behavior.

OServ (Object Server) is written in Java language. And it receives user's classes and holds them. When it receives a request, it executes user's classes that have already been registered. Each static variable is held throughout their requests.

Originally the concept of invocation of service was derived on a model that is called client-server relationship. It had strongly needed for efficient use of hardware resource. Programmer can reduce overhead of procedure to connect to remote server and implements their program easily.

Registration
cd counter
java -classpath ../OServ.jar oserv.ObjectTool sample.CounterServer

cd userclient
java -classpath ../../OServ.jar:./CounterFunction.jar:./ CounterClient clear
Value 0

java -classpath ../../OServ.jar:./CounterFunction.jar:./ CounterClient2 clear
(see debug messages)

CounterClient2 class is a sample for specifying user's property file. Reader can confirm content of property file by extracting oserv.properties file as like follows,

cd counter/userclient
mkdir oserv
cd oserv
jar xvf ../CounterFunction.jar oserv.properties

CounterClient2 contains code in source as like follows,

```
cs = new CounterServer();
try {
      cs.readPropertyFromJar("CounterFunction.jar");
} catch (Exception exception_object) {
      System.out.println("CounterFunction");
       return;
}
```

readPropertyFromJar() method search oserv.properties file in jar file that is specified in CLASSPATH variable. The oserv.properties involves oserv.debug.level property with 5. Reader can see debug messages when they execute this sample CounterClient2.

Appendix B. Configuration of forward

An example for single layer service

Client side configuration
oserv.forward.tunnel=1
oserv.request.port=2003
oserv.forward.servers=oserv://127.0.0.1:2013/

Server side configuration
oserv.request.port=2013
oserv.forward.clients=oserv://127.0.0.1:2003/

An example for multi layer service

Client side configuration
oserv.forward.tunnel=1
oserv.request.port=2003
oserv.forward.servers=oserv://127.0.0.1:2013/
oserv.forward.clients=oserv://127.0.0.1:2023/

Server side configuration
oserv.request.port=2013
oserv.forward.clients=oserv://127.0.0.1:2003/

Second client side configuration
oserv.forward.tunnel=2
oserv.request.port=2023
oserv.object.servers=oserv://127.0.0.1:2003/

Appendix C. ObjectDirectory

ODIR package is developed as sample of OServ (ObjectServer). It contains features for manipulation of file. It indicates that OServ has an advantage for building distribution system. Reader can know how to operate and how easy they use.

```
# cd ODIR

# For local execution.
# mkdir WORK
# cd WORK

# java -classpath ../odir.jar odir.Register ttanaka 'password'
# ls
ttanaka     user_database
# java -classpath ../odir.jar odir.User ttanaka 'password'
Entry
ttanaka
# java -classpath ../odir.jar odir.Register koizumi 'koizumi33'
# java -classpath ../odir.jar odir.USer ttanaka 'password'
Entry
ttanaka
koizumi
# java -classpath ../odir.jar odir.Erase ttanaka 'password'
# java -classpath ../odir.jar odir.User koizumi 'koizumi33'
Entry
koizumi

# For distributed execution
# cd ODIR
# java -classpath ../OServ.jar oserv.ObjectTool odir.DirectoryServer
# java -classpath ../OServ.jar oserv.ObjectTool -l
Entry
odir.DirectoryServer
# mkdir DIST
# cd DIST
# jar xvf ../DirectoryServer.jar odir/DirectoryService.class
# java -classpath ../../OServ.jar oserv.ObjectGenerator DirectoryService
odir.DirectoryServer
# javac -classpath ../../OServ.jar:./ odir/DirectoryServer.java
# java -classpath ../../OServ.jar:./:../odir.jar odir.Register ttanaka 'password'
# ls
```

odir

java -classpath ../odir.jar odir.User ttanaka 'password'
Entry
ttanaka

Command Reference
ODIR package has several commands for operation. This section describes syntax of command line.

cd ODIR
cp ../OServ.jar .

How to register Server module for directory service.
java -classpath ./OServ.jar oserv.ObjectTool odir.DirectoryServer

How to create user account
java -classpath ./OServ.jar:./client:./odir.jar odir.Register ttanaka 'password'

How to make user's account
java -classpath ../../ odir.User ttanaka 'password'

How to delete user's account
java -classpath ../../ odir.Erase ttanaka 'password'

How to make directory
java -classpath ../../ odir.Directory odir://localhost/TEST ttanaka 'password'

How to get a list of content in directory
java -classpath ../../ odir.List odir://localhost/TEST ttanaka 'password'

How to delete a directory
java -classpath ../../ odir.Delete odir://localhost/TEST ttanaka 'password'

How to save a file
java -classpath ../../ odir.Put ttanaka 'password' Check.class odir://localhost/Check.class

How to retrieve a file
java -classpath ../../ odir.Get ttanaka 'password' odir://localhost/Check.class Check.class

How to remove a stored file
java -classpath ../../ odir.Remove odir://localhost/Check.class ttanaka 'password'
(You can specify wildcard character for file group that consist of same prefix name. The URI path in the syntax is 'odir://localhost/PP.*'.)

How to execute a stored file
java -classpath ../../ odir.Execute ttanaka 'password' odir://localhost:2001/Check.class Check outValue1 2 3 2.2 3.3

How to execute a stored file
java -classpath ../../ odir.Execute ttanaka 'password' odir://localhost/Check.jar Check outValue1 1 5 2.4 3.4