

www.PhOSCo.com

# ***Trial Server User Guide***

*Version 1.00 Beta Release*



**Guillemot Design Ltd**

*Pharma Open Source Community:*  
**PhOSCo Clinical Trials**

# Trial Server User Guide

*by Mike Calder for Guillemot Design Limited*

<b>Version</b>	<b>Date</b>	<b>Revision History</b>	<b>Author</b>	<b>QA</b>
1.00	27/07/2000	Beta Release	MACS	CAC

While we have taken great care to ensure that this document provides an accurate description of the subject matter, as software changes over time and as errors do occur in software, you must not depend on any described function without checking that it performs correctly in your environment and circumstances of use.

This document is provided for the guidance of PhOSCo subscribers only. As the installation, customisation, and mode of use of any materials provided by Guillemot Design Ltd are entirely outside our control, we can take no responsibility for the accuracy or appropriateness of any information or advice contained in this document with regard to any particular situation and will not be liable for any loss arising out of use of this document or the software it describes.

Users are recommended to fully analyse any situation or to take appropriate advice while implementing any system.

"Pharma Open Source Community" and "PhOSCo" are trade marks of Guillemot Design Ltd.

This manual is copyright © 2000 Guillemot Design Ltd. It may be freely copied as a whole without changes or additions, but permission of the copyright holders is required for use of extracts other than for purposes of review or research.

# Table of Contents

Trial Server User Guide.....	1
Trial Server.....	2
Installation.....	4
Initialisation.....	4
Start-up Parameter File.....	5
Starting up.....	6
Trial Server Commands.....	9
accept off.....	9
accept on.....	9
delsf n.....	9
delsmx n.....	9
list.....	9
listsf.....	9
listsmx.....	9
listsites.....	10
listsubj.....	10
locking off.....	10
locking on.....	10
qnum.....	10
recalc.....	10
sf site seq ctlname.....	10
smx site n n n n.....	11
stop.....	12
term n.....	13
verbose off.....	13
verbose on.....	13
Error Messages.....	14
Security.....	15
Log Files.....	17
Other Files.....	17

## **Trial Server User Guide**

The PhOSCo Clinical Trials Trial Server User Guide is intended as an introduction to the principal elements of PhOSCo Trial Server, how to run the program, and as a reference to the console commands of Trial Server.

## **Trial Server**

Trial Server is the application program within PhOSCo Clinical Trials that is used to communicate with remote computers running Trial Recorder and synchronise the data on the database local to Trial Server with those on the remote machines.

This server requires a console to itself, it is not a daemon. On start-up it spawns a Command Input thread which blocks until text is entered via stdin.

The Trial Server mainline is basically a loop waiting for requests from clients on a server socket. When one is received, a new service thread is created with a separate session socket, and communications with the client deputised to that thread.

Client session requests are initiated by the Communications thread of Trial Recorder, which periodically polls to see if there is a connection via its TCP/IP environment to the Trial Server defined in the Trial metadata for the site it is running. If the thread finds a connection to the defined port, and Trial Server is accepting requests on that port, a session start is requested.

The mainline also owns a command line thread. Commands from that and messages from the service threads are executed immediately by a routine here called from the client service threads. Periodically after a system-defined time the server socket wait is interrupted to allow stop commands to start closedown.

Communications to higher level servers is handled by a single communications forward thread, which is created immediately after the command line thread and before waiting for service requests from clients. This thread behaves much as the Communications thread in Trial Recorder does in talking to Trial Server.

The Server as a whole produces a new log file per execution of the Server. This file should be renamed or copied on Server close if contents are required to be kept. Logging may be "verbose on" or "verbose off". "Verbose on" enables maximum traceability and is strongly recommended.

The program has been written so that it should be possible to leave it running unattended. If a failure in a communications session occurs, the problem is logged and the session and service thread terminated. Normally a client site will retry later and recover automatically from the error.

*Trial Server User Guide - Version 1.00 Beta Release*

Synchronisation is done transaction by transaction, with each client site keeping track of where it has got to, and storing that in the database. A record is only logged as being synchronised at the Server when a message to that fact is received at the client. So, if there are any communications or other failures, after restarting the client can take up exactly where it left off, without any need for recovery action or rollback.

In the event of exceptional circumstances, facilities are provided to download test utilities and scripts, fetch client log files and test results, and reset the client transmission log records from the central point, so that recovery action can be performed by central site personnel without involving the remote user in any IT procedures.

## **Installation**

Installation of Trial Server, the database, and the start-up parameter file may have been done for you.

If not, consult your distribution documentation to get instructions on how to install the Trial Server software itself, and the prerequisite Java engine.

Besides Java and Trial Server itself, you must provide an SQL database engine. Any SQL engine that provides ANSI compatible SQL via JDBC will work with Trial Recorder, but you must make sure that either the SQL software provided includes either a JDBC driver. Make sure you obtain at least a level 1.2 installation.

Note that ODBC will not work properly, and you should not attempt to use the JDBC-ODBC bridge. This is because the ODBC route cannot be guaranteed to work properly with multi-threaded applications such as Trial Server.

Once your SQL engine is installed, set up the database as outlined in the manual "PhOSCo Clinical Trials Database".

See the later section for the contents of the start-up parameter file "SERVER.DAT".

## **Initialisation**

Please note that before Trial Server is run for the first time for a trial, the database must contain the metadata for the Trial that this Server is a Server for, and the Server database must be initialised with the password hashcodes for the site(s) it serves.

The Trial metadata can probably be most easily loaded on the local database using the "JCTInitialLoad" utility. See the Trial Recorder User Guide for details of this utility and its operation.

The password hashcodes can be initialised by opening Trial Recorder against the Trial Server database with the TSF containing the site and user definitions.

## Start-up Parameter File

Before Trial Server can be run, a start-up parameter file must be made available for it under the name of **SERVER.DAT**. This must be present in the current directory when Trial Server starts.

This file identifies the Trial being used whose metadata must be found in the database, and the identifier of the Server, and so on. The contents of a sample file would look like this:

```
// This is a sample JCT Server control file
//
// It defines the trial and sites that this server handles.
// The trial as listed must exist in the metadata database, and sites
// listed for the trial must exist in the metadata for that trial.
//
// Lines must start with a keyword or a comment identifier - no leading
// whitespace. Tokens in trial or site record lines must be separated
// by at least one space character; more whitespace than this is allowed
// and ignored.
//
// Comments are single lines with the first non-blank token // or /* -
// multi-line comments must start with one of these tokens on each line.
//
//
// The trial record consist of the keyword "trial" followed by
//
//     the numeric change control level for the active trial
//     the name of the trial.
//
Trial 0 PhOSCo Sample Trial
//
// The name of this site which is acting as a server is defined in
// a "ServerSite" record.
//
ServerSite Trial HQ
//
//
Locking yes
//
//
// The Site records for that trial follow:
// one site record for each site which sends data to this server,
// which consist of the keyword "Site" followed by the site name.
//
Site Derby Royal
//
//
// If the server is also a client to a higher level server, then under
// the trial record in which this server is a client, the site to
// which records should be sent must be identified by a ClientTo record.
// Any server identified by a ClientTo record should not also be
// identified in a Site record, as the ClientTo processing will handle
// any data originating at the higher level site.
//
// ClientTo records consist of the keyword "ClientTo " followed by the
// site name of the server which this server feeds.
//
//
ClientTo Trial HQ
//
//
```

## Starting up

Trial Server is a console application, and does not require a GUI environment. It should be invoked from a command-line console. Trial Server is not a daemon, and will use the console invoked from for reporting log messages and taking command input.

The command to start Trial Server will vary according to your local installation (it may be encapsulated in a script), but as a command line invocation the JCTServer class which is the main entry point for Trial Server requires three parameters:

*dbname uid pwd*

where:

*dbname* is the symbolic name used to refer to the local PhOSCo Clinical database to be synchronised by Trial Server,

*uid* is the database userid, and

*pwd* is the database password.

These may be required as options to the command required to start Trial Server; *uid* and *pwd* at least should be, as it is insecure to store these as parameters in a script or command file.

The program should respond with the following messages, provided the database is available:

```
2000-07-18 10:15:42.37...PhOSCo Trial Server Version 1.0
```

```
PhOSCo Trial Server:
```

```
2000-07-18 10:15:42.42...Copyright 1999, 2000 Guillemot Design Ltd.
```

```
PhOSCo Trial Server:
```

```
PhOSCo Trial Server: Verbose 0N
```

```
Supports Transactions
```

```
Opened Metadata Database for user xxx
```

```
2000-07-18 10:15:48.25...Serving Trial PhOSCo Sample Trial
```

```
PhOSCo Trial Server:
```

```
2000-07-18 10:15:48.25...Server site name Trial HQ
```

## *Trial Server User Guide - Version 1.00 Beta Release*

Ph0SCo Trial Server:

2000-07-18 10:15:48.25... Locking on

Ph0SCo Trial Server:

2000-07-18 10:15:48.25... Serving site Derby Royal

Ph0SCo Trial Server:

2000-07-18 10:15:48.3... Control File Read.

Ph0SCo Trial Server:

? for command list

Any other output seen will usually be because of either a faulty installation, or messages indicating that the database is not available for various reasons. If you see any of these. Fix the appropriate problem and retry.

Every message from Trial Server to the screen that is prefixed by a timestamp will normally also be written to a log file.

Entering a single ? Character will give you the list of available commands:

*Trial Server User Guide - Version 1.00 Beta Release*

?

2000-07-18 10:27:16.63...Command > ?

Ph0SCo Trial Server:

Commands:

accept off            Stop accepting connections  
accept on            Start accepting connections  
delsf n              Delete the nth outstanding SF command  
delsmx n             Delete the nth outstanding SMX command  
list                 List active sockets  
listsf               List outstanding Send File commands  
listsmx              List outstanding SMX commands  
listsites            List valid sites in trial  
listsubj             List subjects needing status recal c  
locking off          Switch locking off  
locking on           Switch locking on  
qnum                 Query number of subject a/w status recal c  
recalc               Recalculate subject statuses  
SF site seq ctlname Add file to list to be sent to site  
    site = site number  
    seq = sequence number (unique within site)  
    ctlname = control file name in XMT directory  
SMX site n n n n n Update site "so far" numbers  
    site = site number  
    n n n n n = lg, nt, sr, su, tx  
stop                 Close Down after active sockets finish  
term n               Terminate socket n  
verbose off          No Verbose error messages  
verbose on           Verbose error messages  
?                    Display command list

Ph0SCo Trial Server:

## **Trial Server Commands**

### ***accept off***

Entering this command will stop the Trial Server mainline from accepting new service requests from clients, and as existing session threads close, the server will cease to perform update activity.

The reason for this capability is so that Status Recalculation can be performed periodically. Status Recalculation is not performed concurrently with database synchronisation because of the likelihood of high rates of lock contention. If Clients are held off contact with the local Server database, recalculation can be run without Clients or the batch validation process itself being affected by locking conflicts.

The Status Recalculation is initiated by the "**recalc** c" command (q.v.), and completion of the recalc process should be followed by switching accept back on.

### ***accept on***

See "**accept off**" and "**recalc** c".

### ***delsf n***

Delete the nth SF (SendFile) stored command - see "**sf**".

### ***delsmx n***

Delete the nth SMX (SiteMatrix Xmit) stored command - see "**smx**".

### ***list***

List the currently active sessions and the sockets they are on.

### ***listsf***

List the currently stored SendFile commands - see "**sf**".

### ***listsmx***

List the currently stored SiteMatrix Xmit commands - see "**smx**".

### ***listsites***

List the site names and numbers that are valid for this Server.

### ***listsubj***

List the Subjects and key fields for pages in current need of status recalculation.

### ***locking off***

Turn off database locking (the default is set by the Trial Server parameter file).

### ***locking on***

Turn on database locking (the default is set by the Trial Server parameter file).

### ***qnum***

Query the number of subject key sequences awaiting status recalculation.

### ***recalc***

Perform Status Recalculation. This command invokes a Batch Validator on the current Trial, and recalculates the status for all the subjects and pages for which updates have been received.

Trial Server stores these key sequences in a file between Trial Server sessions.

Which recalculation is in course, it is suggested that it is appropriate to have shut down client access to the local database using the "**accept off**" command and that once recalculation is complete, access should be re-provided by using the "**access on**" command.

### ***sf site seq ctlname***

Define a SendFile command and stores it.

This command controls the transfer of files required to be sent to the client computer from the Server computer or vice versa. The command is stored,

and executed during the next communications session with the defined client site during the "Urgent Data" sequence. All commands not executed during the current Trial Server session are saved to an external file when Trial Server is closed and re-read when the next Trial Server session is started.

File transfer is controlled by a control file, whose name is defined in the command. This file must exist in a subdirectory called "xmit" under the current directory. The control file must consist of three lines of ASCII text.

The first line in the control file must be the fully qualified path and file name of the source file to be sent, on either the Server or Client machine.

The second line in the control file must be the fully qualified path and file name of the required destination on either the Server or Client machine.

The third line in the control file must be a series of one or more commands all on a single line, delimited by single ';' characters. The only commands so far defined are:

**"SendToClient"** - defines that the file transfer is to be from Server to Client,

**"GetFromClient"** - defines that the file transfer is to be from Client to Server.

These two are mutually exclusive and either of these must be terminated with a single ';' character with no whitespace.

The command itself has three parameters, the first ("*site*") being the number of the site to or from which the transfer is to take place. You can find these site numbers using the "**listsites**" command.

The second parameter ("*seq*") is a unique arbitrary integer number used to identify this command. You can use any integer number here as long as it is unique within the list of all current SF stored commands.

The third parameter is the file name of the control file, which must be stored in the xmit subdirectory under the current directory.

### ***smx site n n n n***

Define a Site Matrix Xmit command and store it.

This command defines the set of numbers to be reset in the Client site Site

Matrix database table, which in turn control the transactions to be sent from the Client to the Server to synchronise the databases. The command is stored, and executed during the next communications session with the defined Client site during the "Urgent Data" sequence. All commands not executed during the current Trial Server session are saved to an external file when Trial Server is closed and re-read when the next Trial Server session is started.

This command should never need to be used in normal processing, and should only be used during disaster recovery by someone who is aware of all the implications of using it.

The command parameters are the Client site number - you can find these numbers using the "**listsites**" command - followed by five integer numbers. These numbers are used to replace the current "so far" numbers in the record in the Client SiteMatrix table which controls transmission to this Server. The numbers refer to the audit trail transaction sequence numbers for the Log, Note, Subject Resource, Subject, and Transaction Audit Record tables respectively.

The effect of this command is to reset the transmission position for synchronisation transactions between the Client and Server, and should only be done with a knowledge of the current relative states of the two databases. The most likely use is to reset all values to zero in the event of a complete loss of site data, though intermediate values can be determined for backup checkpoints by appropriate inspection and analysis of the log files.

## **stop**

Stop the Server and close down.

Server close down does not necessarily happen immediately. The effect of this command is to stop any acceptance of new Client sessions, and wait for existing Client sessions to terminate. When all existing Client sessions have closed, the Server will itself close.

**Ph0SCo Trial Server:**

**stop**

**2000-07-18 12:31:34.03...Server Stop request received. Going for closedown.**

Ph0SCo Trial Server:

2000-07-18 12:31:36.5... There are 0 services currently active.

Ph0SCo Trial Server:

2000-07-18 12:31:36.5... Closedown complete.

If it is necessary to perform an urgent close down, this should be done in a controlled manner using the "**term**" command. This can be used after a "**stop**" command to close existing Client sessions immediately, and when the last has been closed, the Server will close down.

### ***term n***

This command is used to close a Client communications session immediately, without waiting for it to complete. This can be done perfectly safely at any time, as the system will restart at the first non-completed transaction the next time a session is established between this Client and Server.

The parameter is the integer socket index number identifying the session to be closed. The sequence of socket index numbers starts at zero, and is the same as that displayed using the "**list**" command.

You will be asked to confirm that the socket is to be closed, and entering "**yes**" will close the session, and any other response will not.

### ***verbose off***

This command will switch off verbose messaging mode and vastly reduce the number of messages in the log file. Most of these are extremely informative and useful in the case of network problems, and it is recommended that you keep verbose mode on in most cases.

### ***verbose on***

This command will switch on verbose messaging mode, the recommended level.

## **Error Messages**

Besides start-up messages indicating that the Server couldn't connect to the database, there are a few environmental error messages which may appear on the console. These are repeated to the console and to the Log File:

**"Couldn't create server socket"**. Probably means that network is incorrectly installed. This is a severe error and the program will terminate immediately without attempting anything further.

**"Couldn't set server socket timeout"**. Probably means that something is wrong with the network. This is a severe error and the program will terminate immediately without attempting anything further.

**"Exception on thread create. Halting."** Probably means that system resources are overloaded. No further attempt will be made to accept socket connections, and the number of exceptions will fall as clients disconnect, when the program should be closed and restarted, perhaps after fixing any problem causing the overload.

**"I\O Exception on socket connect. Halting."** Probably due to network failure. Again no further attempt will be made to accept new socket connections. Fix network problem, close and restart program.

**"Unable to open log file - logging inoperative."** As it says. Without a log trace, you may wish to close the server down and find out why (disk full, log file locked by another process?). The server will continue, but you probably shouldn't.

Any other error messages will be logged to the DB.LOG file if they are database exceptions, as well as appearing on the console.

## **Security**

The current code here has a restricted amount of security, and depends largely on its environment for protection. It should be behind a firewall which for the server machine only allows known IP addresses through to the defined port, and which has if possible some mechanism to guard against IP address spoofing and port storming attacks.

The fact that the server uses a proprietary protocol, not one of the standard Internet protocols such as HTTP, means that general attacks and protocol trapdoors will not work against Trial Server.

The code is specifically written to avoid buffer overrun attacks.

There is the potential for use of cryptographic techniques within the server to provide a higher level of security, but these are not made generally visible for obvious reasons.

The principal measures taken inside Trial Server are:

- A stub is provided for table lookup of known valid network addresses and vetting of requests from clients provided with dynamically assigned IP addresses by Internet Service Providers.
- Standard Java Stream handling and I/O Exception handling is used against potential buffer overrun attacks.
- Duplicate socket requests from the same IP Address are denied and both duplicates thrown off (one potential denial of service attack).
- Reconnection attempts from the same IP Address within a system defined minimum time are refused.
- Any requester not providing as the first request a valid IDENT transaction containing valid references to a trial and site in the metadata database of the server is thrown off.
- Any IDENT purporting to be from a site but not from the IP Address defined for that site within the trial metadata is refused.
- Any IDENT from a site not defined as being the responsibility of this server in the metadata for the trial is thrown off.
- Any non-recognised or out of sequence transaction requests cause the requester to be thrown off.

*Trial Server User Guide - Version 1.00 Beta Release*

- All potential security exposures are timestamped and logged to the console and a security log file for warning, inspection, and study.
- All normal session opens and closes are timestamped and logged to a system log and to the console.
- All exception conditions are timestamped and logged to the console and an exception log.

## Log Files

Trial Server creates several Log Files:

- **TSERVER.LOG**
- **TRANSACT LOG**
- **DB.LOG**

**TSERVER.LOG** is a log file that contains all the messages sent to the console. This file does not carry over from one session to another, and if required for audit or analysis should be copied or renamed at the close of a session.

**TRANSACT.LOG** does get carried over sessions, and consists of all the transactions that have been successfully received at or transmitted from this Server. It contains for each record a timestamp, a record type indicator, and the record data. The record types are:

- T0 - IDENT records received
- T1 - Records received from the Client
- T2 - Records sent from the Server to the Client

**DB.LOG** again gets carried over sessions, and contains a log of significant events from the Database handler. This primarily consists of SQL Exceptions.

## Other Files

Other files created by Trial Server are:

- **SUBJECT.HSH**
- **SF.HSH**
- **SMX.HSH**

**SUBJECT.HSH** is a file used to keep a record between sessions of the subject pages which require status recalculation because of data received at the Server. When a recalculation is performed, the entries in this file are

removed.

**SF.HSH** is a file used to keep a record between session of the sendfile commands that are to be performed when next a particular Client connects to the Server. When the command is successfully performed, the entry is removed from the file.

**SMX.HSH** is a file used to keep a record between session of the site matrix transmit commands that are to be performed when next a particular Client connects to the Server. When the command is successfully performed, the entry is removed from the file.