WORDFAST SERVER USER MANUAL

Version 1.x ~ All rights reserved

Table of Contents

Introduction

Wordfast Server (**WFS**) is a Translation Memory (TM) and Terminology server. It is used by trained professionals in the translation industry, such as LSPs (Language Service Providers) and translation departments.
Client applications include Wordfast PRO, Wordfast Anywhere, Wordfast Classic, and other compatible applications. Appendix 2 describes an API to build a connector to WFS.

Supported platforms

WFS runs as a 32-bit Windows application on any Windows machine that is connected to the Internet. WFS will also run on 64-bit versions of Windows. Windows 2000™, Windows XP™, Windows server™ 2000, 2003, 2008, 2012, Windows Vista™, Windows Seven™, Windows 8™, can be used as physical platforms to run WFS.

Required environment

WFS does not require any third-party database system – it uses its own proprietary database and indexing system. Thus, it is not necessary to own or enable Windows database services or DBMSes like Microsoft SQL server™, ODBC, Oracle™, etc.
For optimal performance of WFS, you should minimize the number of third-party applications and services installed and running on that workstation..

Client-server communication channel

WFS uses TCP-IP to communicate with clients. It uses its own protocol on top of TCP-IP, as well as its own military-strength encryption method. Thus, all WFS needs is a valid Windows socket. IIS or other services do not need to be activated.

Physical requirements

WFS needs only a basic, Windows platform with an internet connection to run.

Hard disk: A minimum of 10 Mbytes is necessary to install and run the application. For the database (Translation Memories and glossaries), reserve three times the expected bare database size to accommodate indexes and temporary files.
RAM: Over 2 Gbytes of RAM.
Processor: Any processor can be used.

Server capacity

One instance of WFS can exploit a database of up to 10,000 Translation Memories (in as many language pairs as whished), totalling up to 1,000,000,000 Translation Units, and serve up to 5,000 clients.

A collection of TMs is a database, a TM is a *table* within the database, and a TU is a *record* within the table.

Most clients will start with one server serving all languages simultaneously, and will probably never outgrow WFS' capacity. However, it is possible to split the load among as many servers as there are language pairs (one TM supports only one language pair), or even one server per TM, so that stellar scalability is achieved.

Database format

WFS' native TM format is the Wordfast Classic (WFC) TM format, which is a tab-delimited text file designed in 1999, as described in Appendix 4 taken from the Wordfast Classic Reference Manual. Wordfast Classic (WFC) TMs can be opened with text editors, like Notepad, or Unicode-compliant word processors, as well as with Excel. Wordfast TMs can be regular ANSI (8-bit) text, or Unicode UTF-16 (both little-endian and big-endian). WFS can import TUs from TMX TMs (all levels of TMX), and from Wordfast TMs. The WFC TM format is arguably the most compact and reliable TM format in the industry. Note that the WFC TM format does not record text formatting information in extenso, because that is known to bloat TMs. It records placeholders for those "tags", which are meant to preserve formatting. In short, the Wordfast format stores formatting information using a symbolic method.

Backup

The WFS database is a collection of discrete TXT files (one TXT file per Translation Memory and one TXT per glossary), plus one single configuration file with a .stats extension. All other files, such as indexes and temporary files can be recreated as needed by the WFS.

Backing up simply means backing up TMs, glossaries, and the single .stats file. Any standard method can be used, including mirroring, replication, RAID strategies, etc. The only requirement is that the backup method accommodates "live" files. If that is not the case, the server should be stopped before files are backed up. Most modern backup utilities can back up live files.

Deployment

WfServer.exe is a stand-alone Windows application. It does not need any installation. It does not need any DLL. WfServer.exe can be dropped in any folder, and started.

WfServer.exe can be declared as a Windows start-up application (launched when Windows starts) so that it launches itself if the system reboots. WfServer.exe cannot be run as a Windows service.

The workstation where WfServer.exe is installed (the server) must have a fixed sub-domain or domain name, or a fixed IP address, where it can be reached. If the server is located inside a Local Area Network (LAN), and is to be used by translators outside the LAN, the administrator must make sure the server can be reached from outside the LAN using a fixed domain name, or a fixed IP number.

If only translators located within the LAN use the server, it is possible to use only the LAN IP number. The Setup > Network tab displays the server's current *local* (LAN) IP number – note that it does not display the server's WAN, or Internet, IP number. That Internet IP number is usually assigned by the LAN administrator. It can also be determined on the workstation by visiting the following link:
http://www.whatismyip.com/

Important note on sessioning for Remote Desktop (RDP) users. If you RDP into a physical server running Windows, be sure that you always access the same session. For example, if using the Microsoft mstsc.exe RDP application, make sure to use the /admin switch (mstsc.exe /admin) to always access the same root session. Create a dedicated mstsc.exe shortcut if needed. Many users, when returning to their server in RDP, actually open another session, and are surprised to not find the WfServer application running there (although it is found as active by remote client applications such as WF PRO or WF Classic), and start another instance of WfServer. That results in two WfServer applications competing for the same IP number in two different Windows sessions, causing major problems. If you do not understand this paragraph please review this issue with an IT professional.
If you RDP and create a new session by accident (you see a brand new Windows session and no WfServer application running, not even in the system tray), you should close that session using the Windows Start button then *Close Session* or *Disconnect*. Simply closing the RDP window may leave the session running.

Licensing

A license is a contractual agreement between Wordfast LLC and the user to use the WFS software within certain limitations. A license is materialized as a *License number* issued by the licensing party - usually Wordfast LLC - in response to a given *Install* *number*. The *Install number* of a WFS installation is found in the Setup/General pane. That *Install number* has to be provided to the licensing party, which in return will provide a license number.

Note that the Install number is valid for a particular hard disk. If the hard disk is formatted (which can happen if Windows is re-installed), or if the hard disk is changed, the install number changes, and a new license number must be obtained at no cost within the duration of the license agreement.

Demo mode

The demo mode is defined by the absence of a valid license. It is indicated by a flashing RED message in the upper right hand corner of the application window. In this mode, WFS does not accept more than 3 simultaneous connections, and is reserved for private use by individuals, such as freelance translators. Translation/localization agencies, administrations, and corporations in general should use the demo mode only to *evaluate* or test WFS, but not for actual production.

Updates & upgrades

Updates and upgrades to the latest version of WfServer are a simple drop-in file replacement. Here is the detailed procedure:

1. Download the latest version from the Wordfast site here: http://www.wordfast.net/WfServer.zip
2. Make sure WfServer.exe (the application) has been halted. Quit WfServer.exe if applicable. Note that if WfServer is not visible, it may be found minimized in the system tray.
3. Rename the existing WfServer.exe file as WfServer.old. Copy or drag-drop the new WfServer.exe file to take the vacant place of the renamed WfServer.exe.

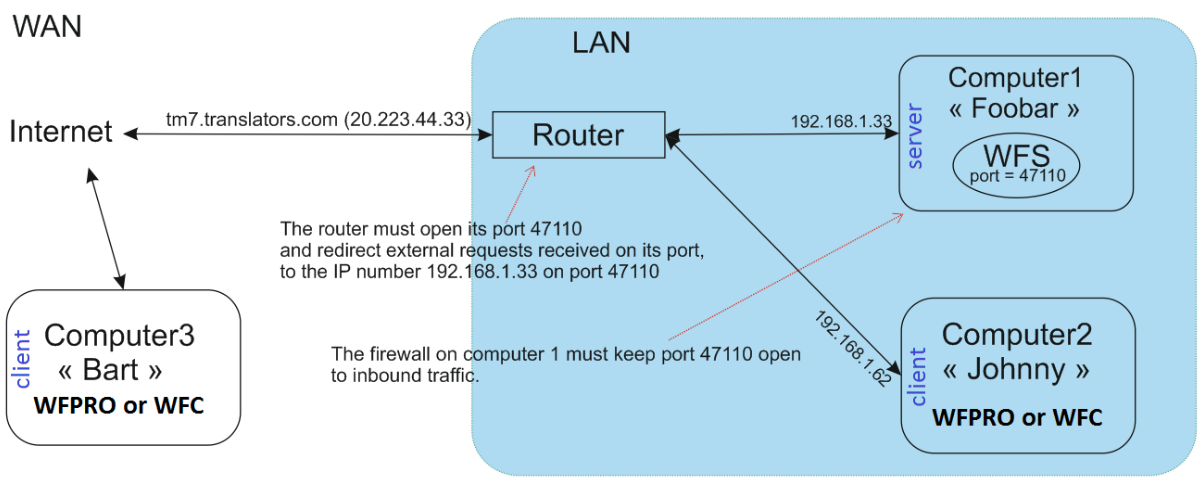
Notes:

- If you are using Wordfast Classic (WFC) to connect to WFS you must also replace the WfServerRelay.exe file with the new one contained in the downloaded WfServer.zip file.
- Your license (and the install number) is not affected by upgrades. Your TMs, glossaries and accounts should remain intact. Nevertheless, we advise backing up the WfServer.stat file located in the same folder as WfServer.exe
- If you were using an older application named WfServerPro.exe, your start file is also named WfServerPro.stat. Make a copy of that file and rename it WfServer.stat.

Setup overview

Bird's-eye view

Here is a snapshot of a typical installation, with one server (Computer1) and two clients, one client in the LAN (intranet) network, and one client out in the web (WAN network).


WAN
LAN
Internet
tm7.translators.com (20.223.44.33)
Router
192.168.1.33
server
Computer1
« Foobar »
WFS
port = 47110
The router must open its port 47110
and redirect external requests received on its port,
to the IP number 192.168.1.33 on port 47110
client
Computer3
« Bart »
WFPRO or WFC
The firewall on computer 1 must keep port 47110 open
to inbound traffic.
192.168.1.62
client
Computer2
« Johnny »
WFPRO or WFC

In the example above,

Computer3 connects to WFS with:

`login:password@tm7.translators.com:47110` , or
`login:password@20.223.44.33:47110`

Computer2 connects to WFS with:

`login:password@Foobar:47110` , or
`login:password@192.168.1.33:47110`

Detailed step-by step instructions

Once WFS is installed, open the application by launching WfServer.exe and uncheck the "Minimize" checkbox in the Setup > Activity section to keep the WFS window visible. This checkbox can be checked during production so that WFS automatically minimizes into the application tray.

Wordfast Server (WFS) exploits TMs. The first step is to tell the server which TMs are being used. TMs must be in either standard Wordfast format (ANSI, UTF-8 or Unicode), or TMX format. The server needs to "reorganize" a TM the first time it is used. Once the reorganization is done, reorganization is no longer needed, unless indexes are lost or damaged in which case the server automatically reorganizes the TM.

The next step is to create accounts that allow clients to connect to WFS. *Clients* are translators using Wordfast Classic, Wordfast Pro or Wordfast Anywhere, and who wish to connect to a TM. Accounts give clients the right to connect to a TM served by WFS. One or more translators (users or clients) can use the same account. Each account has one login name and one password.

For clients to connect to WFS, the WFS server must have a fixed IP, accessible by anyone who connects to it. The relevant port must be opened as well on both client and server sides. Clients need to know the server's IP address, Port # (WFS uses port 47110 by default).

To provide access to a given account, the WFS administrator should provide the client with the following details:

a. The server account Login and Password from step

b. The (sub-)domain name, or the IP Address, of the workstation where WFS is running. The IP can be determined by visiting the following link from the physical server: http://www.whatismyip.com/

c. The Port # from the WFS Setup tab > Network option

d. A WorkGroup ID (if the Public checkbox in the TM tab is not checked). A workgroup ID can be any 10-character code that you choose. The code can contain letters, numbers and/or special characters. Warning: the characters i, l, o, I, and O are not allowed.

The WFS administrator can provide that information in a connection string with the following format:

login:password@DomainName:PortNumber , or
login:password@IPnumber:PortNumber

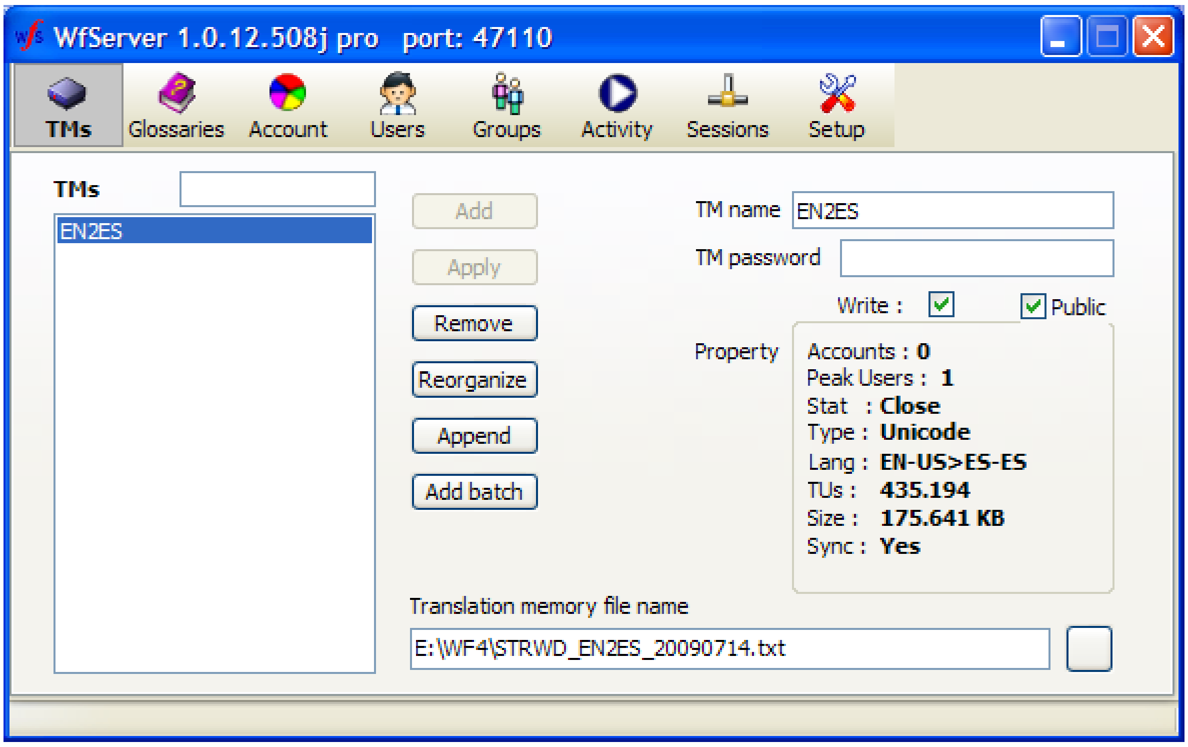
for example,

john:mypass@server1.domain.com:500 , or
john:mypass@167.98.67.1:500

The WFS administrator must have mounted at least one TM, and created at least one <u>account</u> pointing to a TM and/or glossary. Remember that clients connect to <u>accounts</u> (not to TMs). From that point on, activity can start. The "*Server is active*" checkbox must be checked in the "Activity" pane. More TMs and accounts can be added while the server is active.

Many options can be set to administer the server. The options are detailed further below.

The TM tab


WfServer 1.0.12.508j pro port: 47110

TMs
EN2ES

TM name: EN2ES
TM password:
Write : ✓ ✓ Public

Property
Accounts : **0**
Peak Users : **1**
Stat : **Close**
Type : **Unicode**
Lang : **EN-US>ES-ES**
TUs : **435.194**
Size : **175.641 KB**
Sync : **Yes**

Translation memory file name
E:\WF4\STRWD_EN2ES_20090714.txt

Write a "friendly" TM name in the *TM name* textbox. Use short names without accented letters or special characters. Remember that when dealing with complexity, the golden rule is to keep things simple. Keep path (folder) names simple too. WFS supports long path and file names, as well as Unicode folder or file names, but we still advise keeping things simple.

<u>Do not enter a TM password.</u> That field is reserved for rare applications, such as remote administration modules. Protection and security is provided by the password in the Account section of the setup. Leave that field blank, unless you know what you are doing.

- If you wish to <u>**create** a new TM</u>, enter a name in the *TM name* field. Then either:

  ◆ Type a path and file name (complete with a .txt extension) in the *Translation Memory file name file* field on the lower right, or
  ◆ Click on the lower right *Browse* button, navigate to the location where you want to save the TM, type in a name (with a .TXT extension), then click *Save*.

  Click *Add*. You will be prompted to enter a source and a target language. Click *OK*. If prompted to reorganize the TM, click *Yes*.

- If you wish to **add** a pre-existing TM, enter a friendly name in the *TM name* text box, upper right. Click the lower right *Browse* button. With the *Open* dialog box, navigate to locate then open a Wordfast Classic Translation Memory. Those TMs have a .txt extension. Note that after setting up a Wordfast Classic TM in Wordfast Server TM, you can merge it with TMs that have either a Wordfast TXT format, or with TMs that have a TMX format - and usually a .tmx extension.

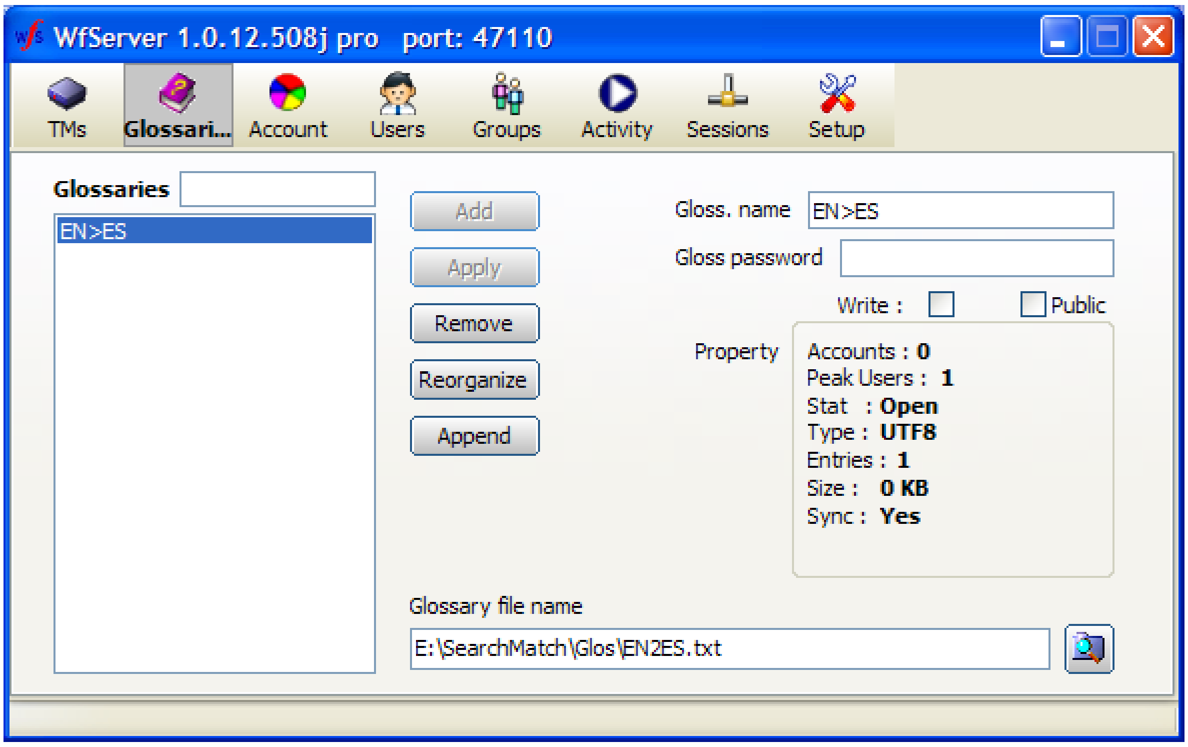
If you allow clients to write to that TM, check the "Write TM" checkbox. Note that you can allow writing to a TM, but deny or authorize write access at the *account* level.

The "Public" checkbox, if checked, indicates that clients do ***not*** need a WorkGroupID to write TUs to the TM. In case of doubt, check that option.

Click the "Apply" button to record the new TM, or any changes, in the list of TMs (it can now be considered that the TM is mounted as a *TM*). Click the Apply button after any change in the setup.

Right-click the list of TMs for a list of actions that can be performed on a particular TM. The most frequently used action is to add Translation Units (TUs) from another TM (the "Append TM" option). These TUs can be taken from a Wordfast TM, or from a TMX TM.

The Glossaries tab



Glossaries
EN>ES

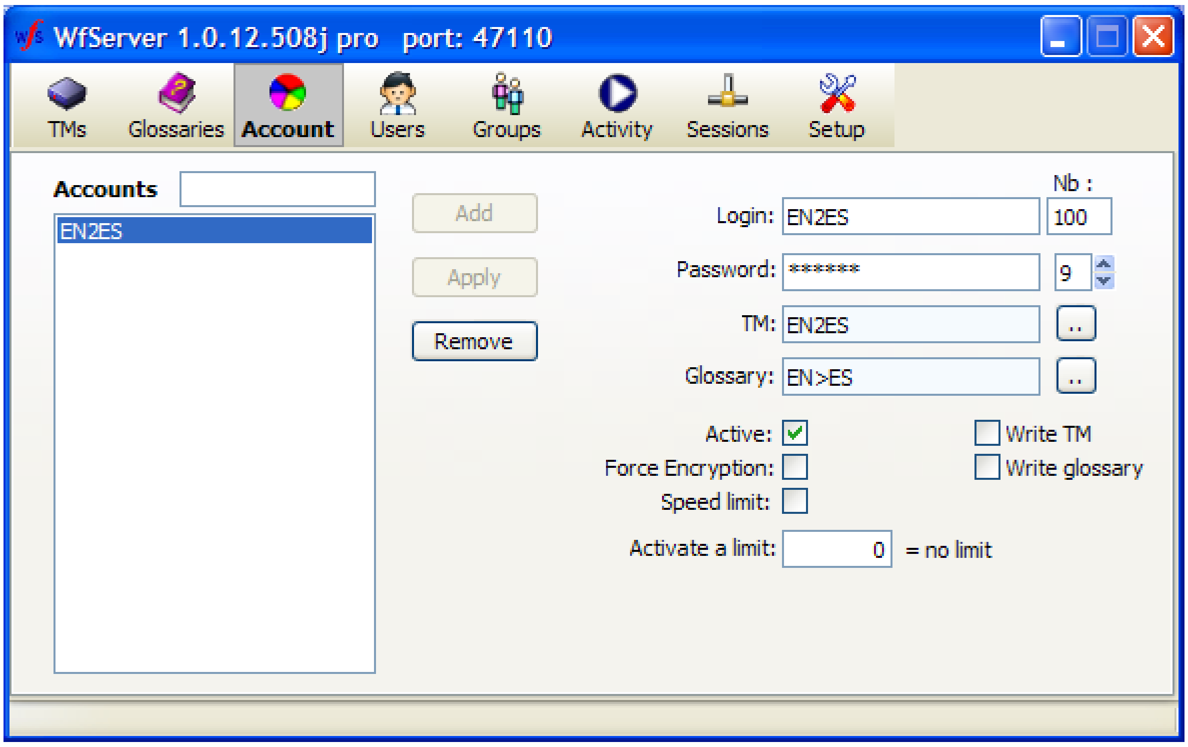
Gloss. name: EN>ES
Gloss password:
Write : ☐ ☐ Public

Property
Accounts : **0**
Peak Users : **1**
Stat : **Open**
Type : **UTF8**
Entries : **1**
Size : **0 KB**
Sync : **Yes**

Glossary file name
E:\SearchMatch\Glos\EN2ES.txt

This tab sets up glossaries, in a manner very similar to a TM.
Glossaries use the standard Wordfast Classic glossary format, in TXT format, ANSI, UTF-8 or Unicode.
Do **not** enter a glossary password unless you know what you are doing. As with the TM tab, note that security is enabled through the use of a password at the account level. A glossary password is only used in rare applications that offer a remote adminsitration mode.
Click the "Apply" button to record the new Glossary, or any changes made to it.

The Accounts tab



Accounts:
EN2ES

Login: EN2ES
Nb : 100
Password: *******
9
TM: EN2ES
Glossary: EN>ES
Active: ✓
Force Encryption: ☐
Speed limit: ☐
Write TM: ☐
Write glossary: ☐
Activate a limit: 0 = no limit

To create an account, write an account name in the *login* textbox. In the *Nb* textbox, enter the maximum number of clients that can use that account simultaneously. Do not leave that setting at 0 (zero), otherwise, no client will be able to connect. Enter a password in the *Password* textbox. Keep the priority level to 10 (highest) if the client is a human translator.

Choose one of the TMs that appear in the list of *TMs*. Choose one glossary that appears in the list of glossaries.

Check the *Write TUs* checkbox if you allow that account to write TUs into the selected TM.
Check *Force encryption* to force the client to use encrypted communication.

Leave the Speed value to its default value (180 requests/**minute**) if the account is to be used by human translators. If the account is used by an automat (for example, a piece of software that can send translation requests at a very high rate), raise that value to 1,000. The value of 180 is high enough to let "human" translators work, but it will block possible hackers that would try to block the server by bombarding it with millions of translation requests.

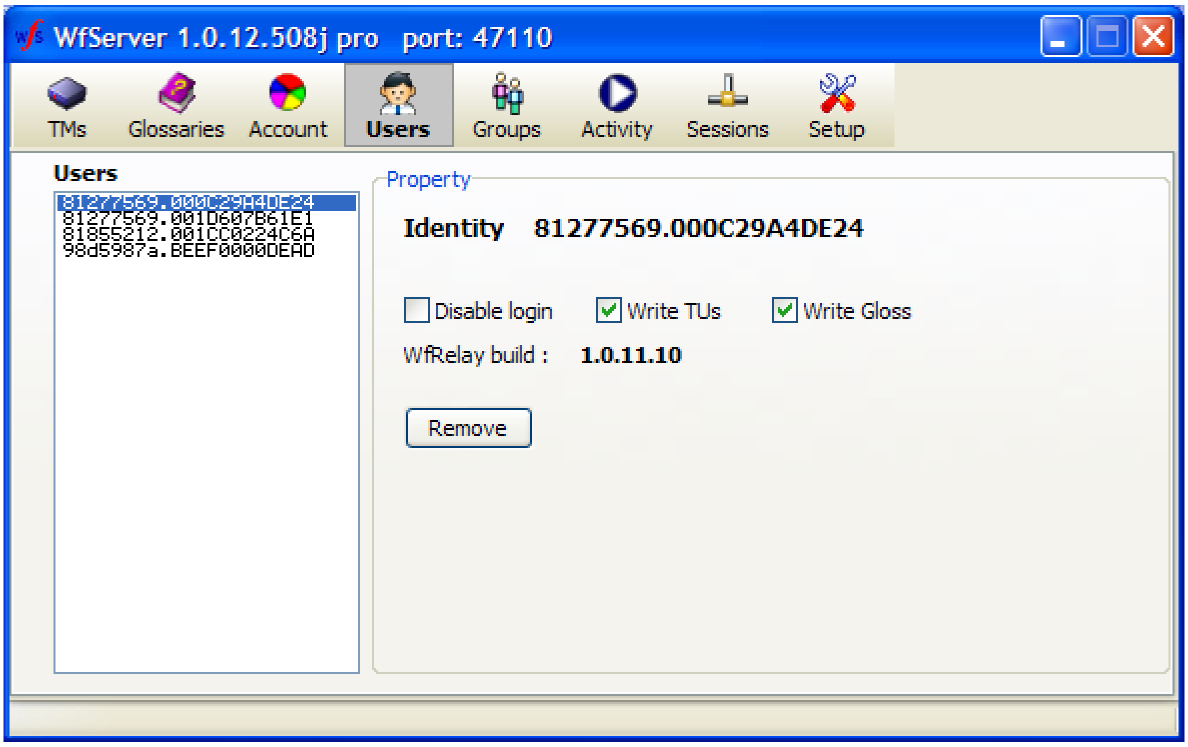
Click the *Add* button. The account has been created.

If you make any changes to an account, use the *Apply* button to enable changes.

Use the *Remove* button to remove an account.

Go to the *Activity* tab and check the *WFS active* or *Server active* checkbox. The server is now running and ready to serve any client using the one account we just set up (*John*), which is connected to one TM (*ES-EN*).

The Users tab



Users

81277569.000C29A4DE24
81277569.001D607B61E1
81855212.001CC0224C6A
98d5987a.BEEF0000DEAD

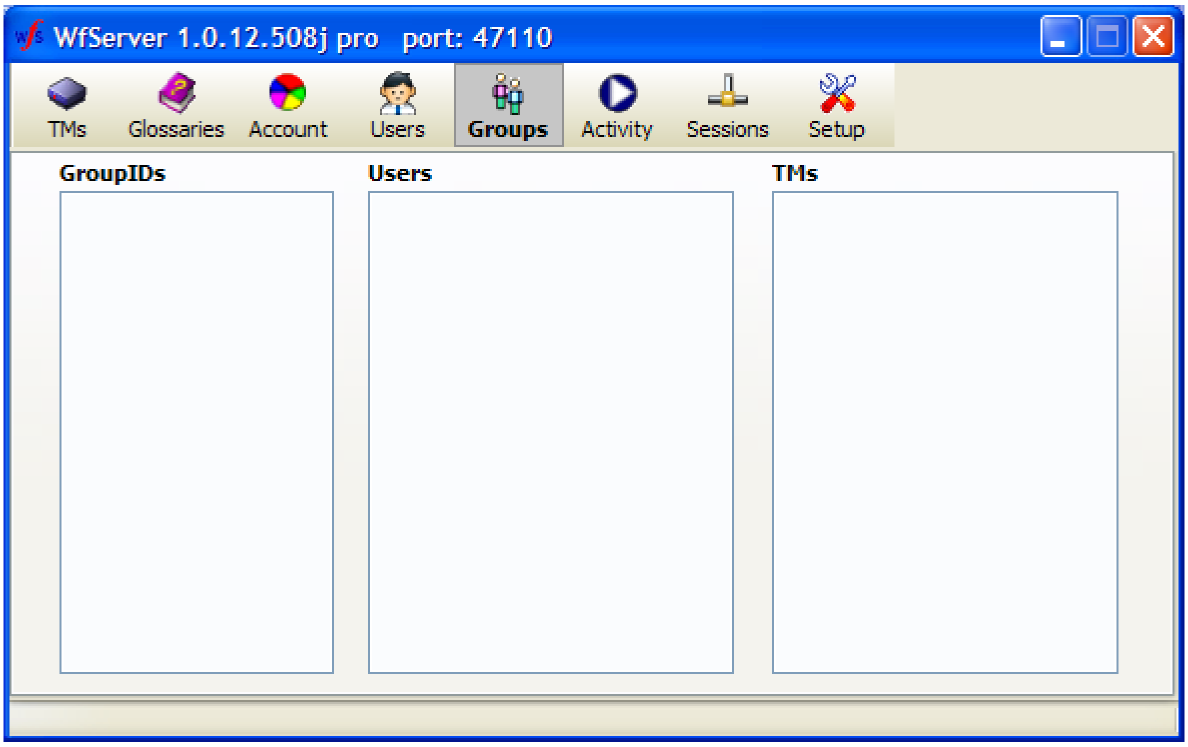
Identity 81277569.000C29A4DE24

☐ Disable login ☑ Write TUs ☑ Write Gloss

WfRelay build : 1.0.11.10

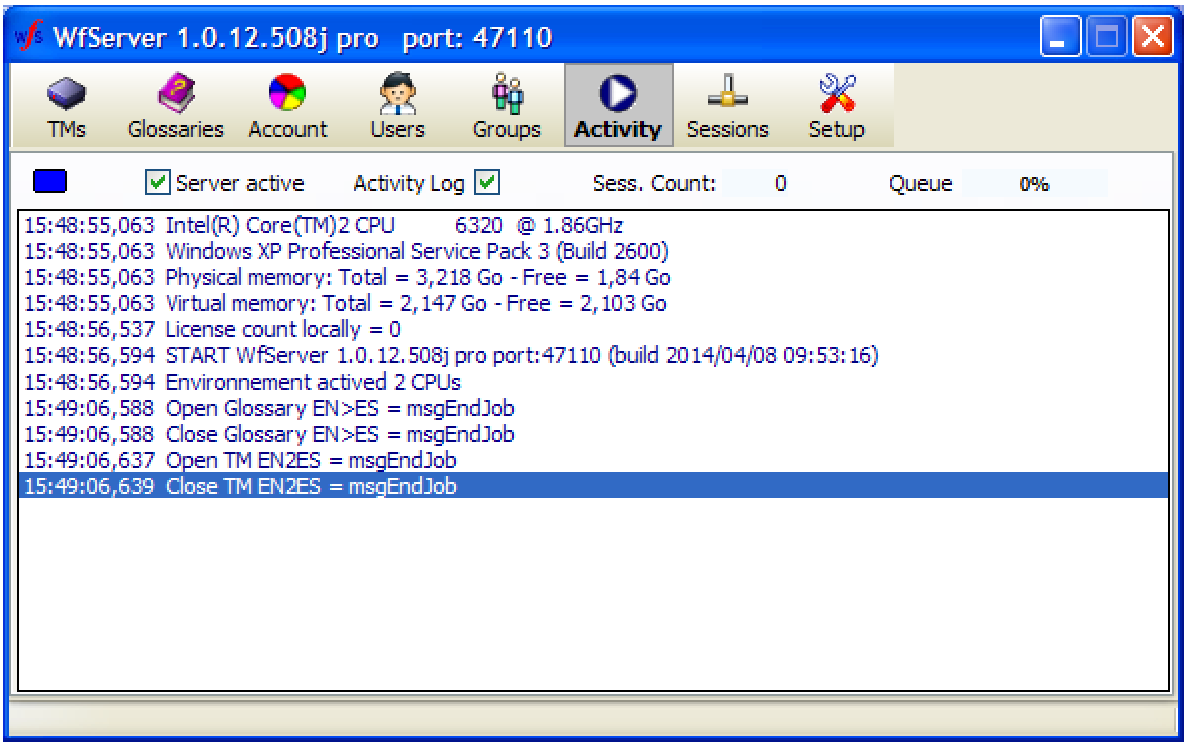
The *Users* tab lets you see which users have been connected to the server. Users are not to be confused with accounts – an account can be used by many users at the same time. A user is any "client" that uses a translation tool which can connect to WFS. WFS registers a few characteristics for each user: its IP number, its DNS name, its MAC address.

The Groups tab



The Groups tab gives the administrator a bird's view of who is currently connected (users), in relation to which accounts and which TMs. Double-clicking any item takes you to the relevant tab and the relevant item for more detail.

The Activity tab



```
15:48:55,063  Intel(R) Core(TM)2 CPU        6320  @ 1.86GHz
15:48:55,063  Windows XP Professional Service Pack 3 (Build 2600)
15:48:55,063  Physical memory: Total = 3,218 Go - Free = 1,84 Go
15:48:55,063  Virtual memory: Total = 2,147 Go - Free = 2,103 Go
15:48:56,537  License count locally = 0
15:48:56,594  START WfServer 1.0.12.508j pro port:47110 (build 2014/04/08 09:53:16)
15:48:56,594  Environnement actived 2 CPUs
15:49:06,588  Open Glossary EN>ES = msgEndJob
15:49:06,588  Close Glossary EN>ES = msgEndJob
15:49:06,637  Open TM EN2ES = msgEndJob
15:49:06,639  Close TM EN2ES = msgEndJob
```

The *Activity* tab lets the administrator monitor server activity.

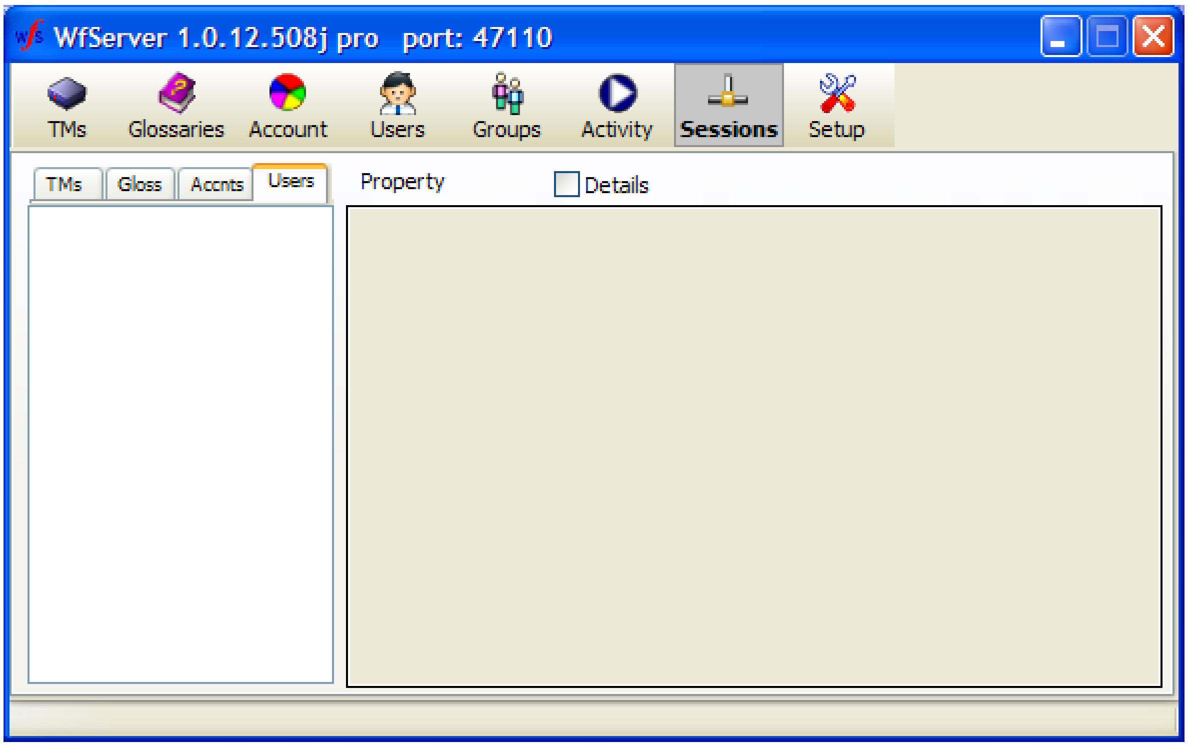
The *Server active* (or *WFS active*) checkbox, when checked, actually turns WFS on (makes it an active server, responding to requests). It should be checked when the server is in use.

The *Activity log* checkbox tells WFS to echo all activity in the textbox. The *Activity* checkbox actually turns itself down if nobody uses the WFS interface (clicks a button or changes a tab) for 10 minutes, to save processor cycles.

The *Sess. Count* number indicates how many users are currently connected to WFS. Users open a session when they first connect to the server. They can close their own session (for example, when quitting their translation tool, the translation tool sends WFS a "Close session" command). A user session is automatically closed if WFS has registered no activity for more than five hours (the number of hours can be set in the Setup > Sessions tab).

The *Queue* progress bar indicates the state of the FIFO queue run by WFS to process incoming requests sent by clients. The priority value associated with an account in the *Accounts* tab sets the account's priority in the queue. It is advised to give human translators a high priority (closer to 10) because they typically send 2 to 10 translation requests per minute, needing immediate response, and a lower priority (closer to 1) to automated processes, like programs that batch-process files, because they can load WFS with thousands of translation requests per minute.

The Sessions tab

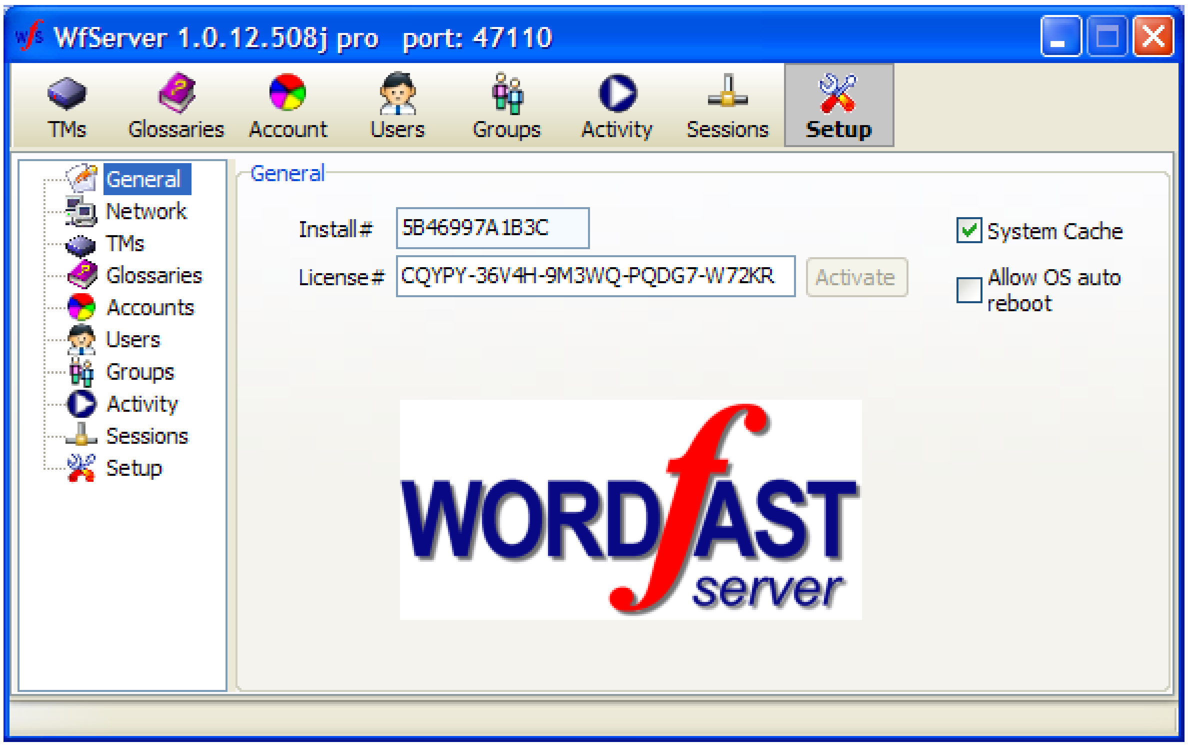


The *Sessions* tab gives an overall view of users, accounts and TMs being used, with detailed stats.

The Setup tab

The Setup tab sets up many WFS options. They are:

General



Install# 5B46997A1B3C

License# CQYPY-36V4H-9M3WQ-PQDG7-W72KR

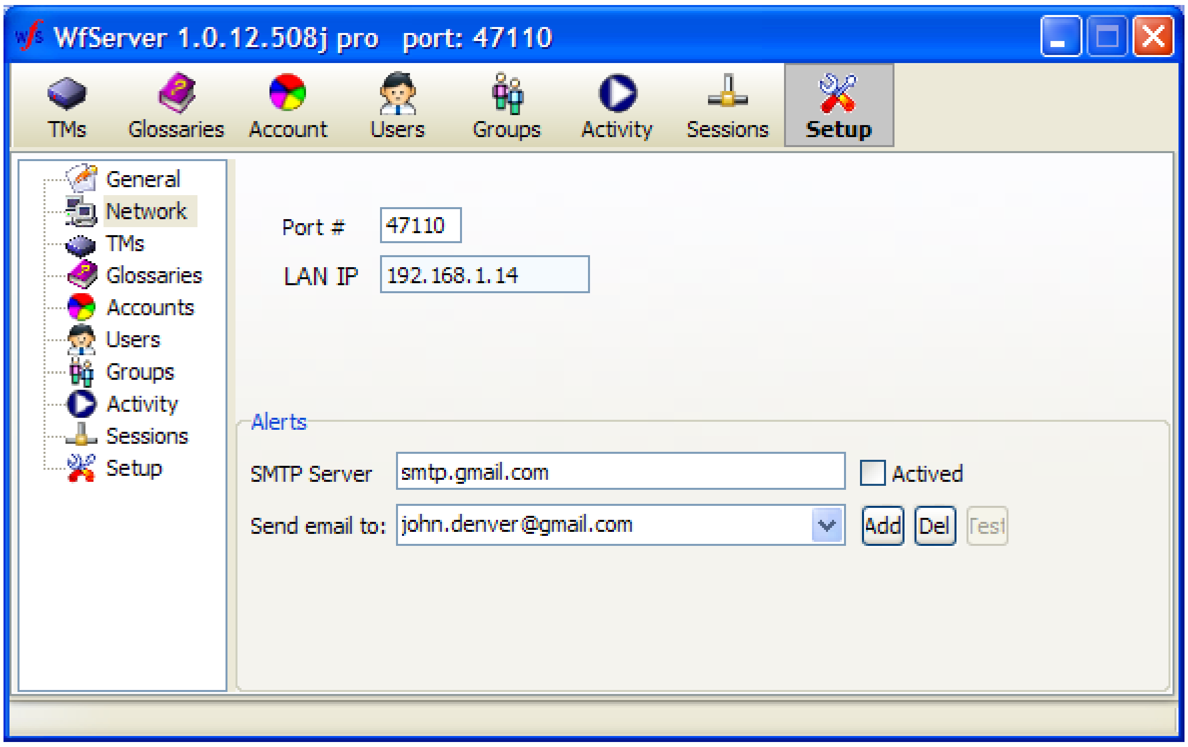
[✓] System Cache

[ ] Allow OS auto reboot

Moral persons with more than 3 employees, such as corporations, translation agencies, institutions, must have a valid license to run WFS in production mode. Without a valid license number, WFS runs in demo mode, which is only allowed for evaluation purposes, or for production use by individual ("freelance") translators working as self-employed individuals. In the demo mode, WFS does not serve more than three "clients" simultaneously. That is the *only* limitation. When you enter a license number in the red textbox, the textbox background turns to white if the license number is correct, and WFS runs in production mode.

Check the *System cache* checkbox to boost hard disk operation performance on regular Windows versions. On Windows *server* versions, the system cache is enabled by default, so it is not necessary to check that option.

Network



Port # 47110
LAN IP 192.168.1.14
SMTP Server smtp.gmail.com
Actived
Send email to: john.denver@gmail.com

By default, WFS uses port 47110 for communication. We advise using that port. The Wordfast client is set to use port 47110 by default, in the absence of a specified port. Port 47110 is not a standard Windows port, and is less likely to be bombarded by hostile pings and pressure. You can use another port number, but then, clients that access WFS must specify the same port number right after the server's IP number in their connection string, as in the following example, where port 500 is forced:

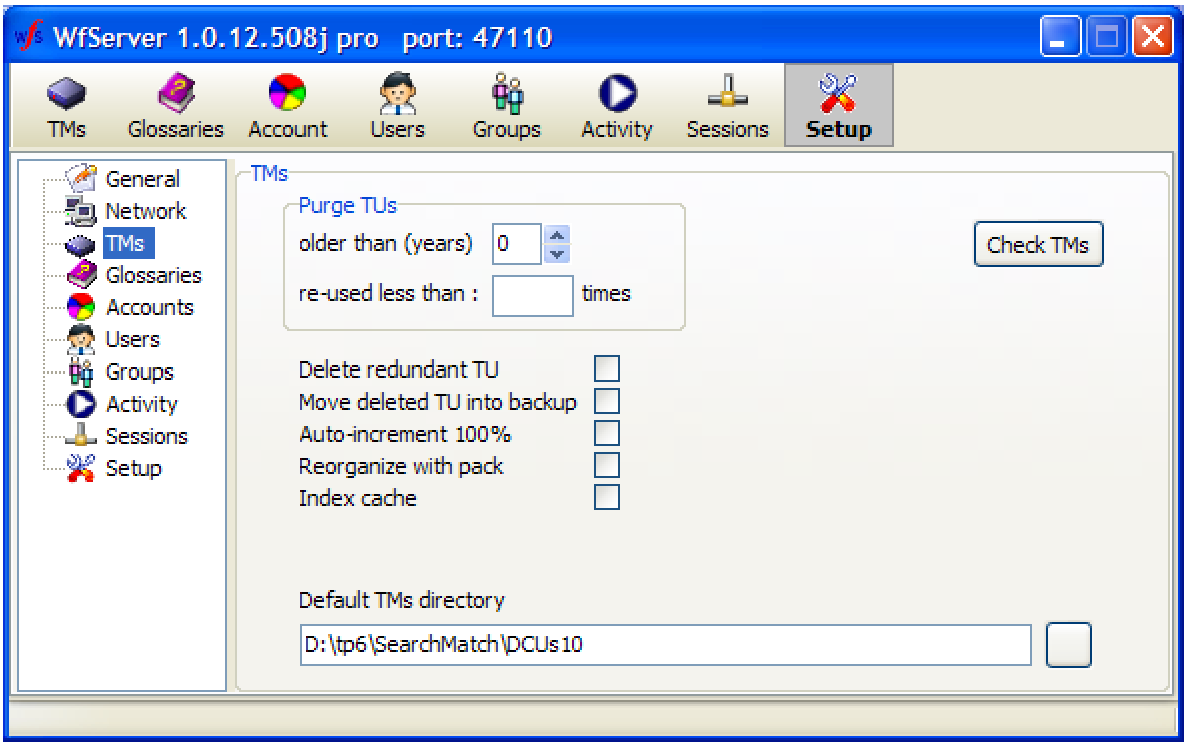
john:mypass@domain.com:500 , or
john:mypass@167.67.78.1:500

If 47110 is used, clients only need to specify:

john:mypass@domain.com , or
john:mypass@167.67.78.1

The WFS administrator can specify an email address (and the relevant SMTP server for mail relay) to receive automatic notifications. Notifications are critical events registered by the server, such as starting up, closing down, or faults.

TMs



TMs

Purge TUs

older than (years) 0

re-used less than : times

Delete redundant TU ☐
Move deleted TU into backup ☐
Auto-increment 100% ☐
Reorganize with pack ☐
Index cache ☐

Default TMs directory

D:\tp6\SearchMatch\DCUs10

The server can automatically delete TUs if:

1. they are older than X years
2. they were re-used less than X times.

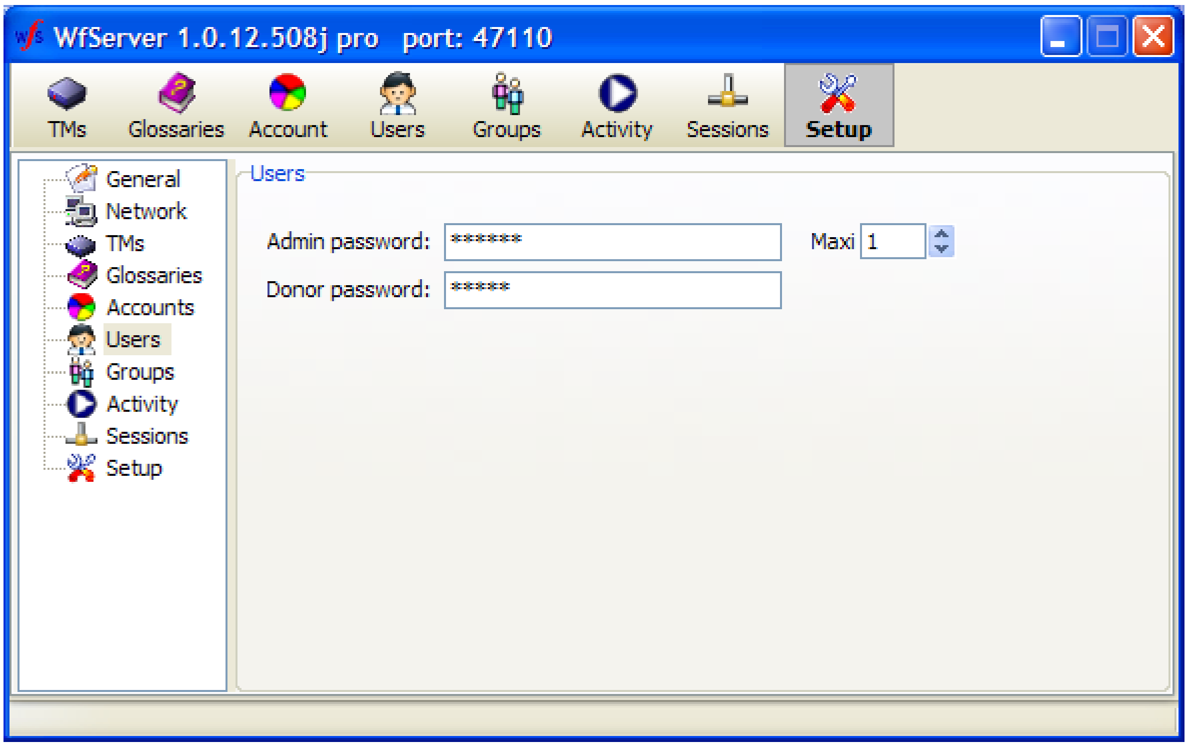
Values 1. and 2. are entered in the *TMs* section of the *Setup* tab. Actual purge is done by right-clicking a TM in the list of TMs in the TM tab, and choosing the *Purge* option.

Check the *Delete redundant TUs* checkbox if WFS must delete *existing* redundant TUs when TUs are being added to the server. Redundant TUs are defined as TUs where source segments are identical. If a different definition of a TU must be used (for instance, redundant TUs are those where both source *and* target segments are identical, or where the degree of matching is above 95%; or where attributes are identical too), the translation *client* should make the decision.

Check the *Auto-increment 100%* checkbox if WFS must increment a TU's "Re-use" counter if it serves a 100% match originating from that TU.

Check the *Reorganise with pack* checkbox if WFS must pack holes during reorganisation. When a TU is deleted, it is filled with whitespace, creating a "hole" in the database. The holes are often reused in real time during additions TUs, but sometimes some holes can not be reused. That is why it is necessary to remove the holes during the reorganization.

Users



Users

Admin password: *******

Maxi 1

Donor password: ******

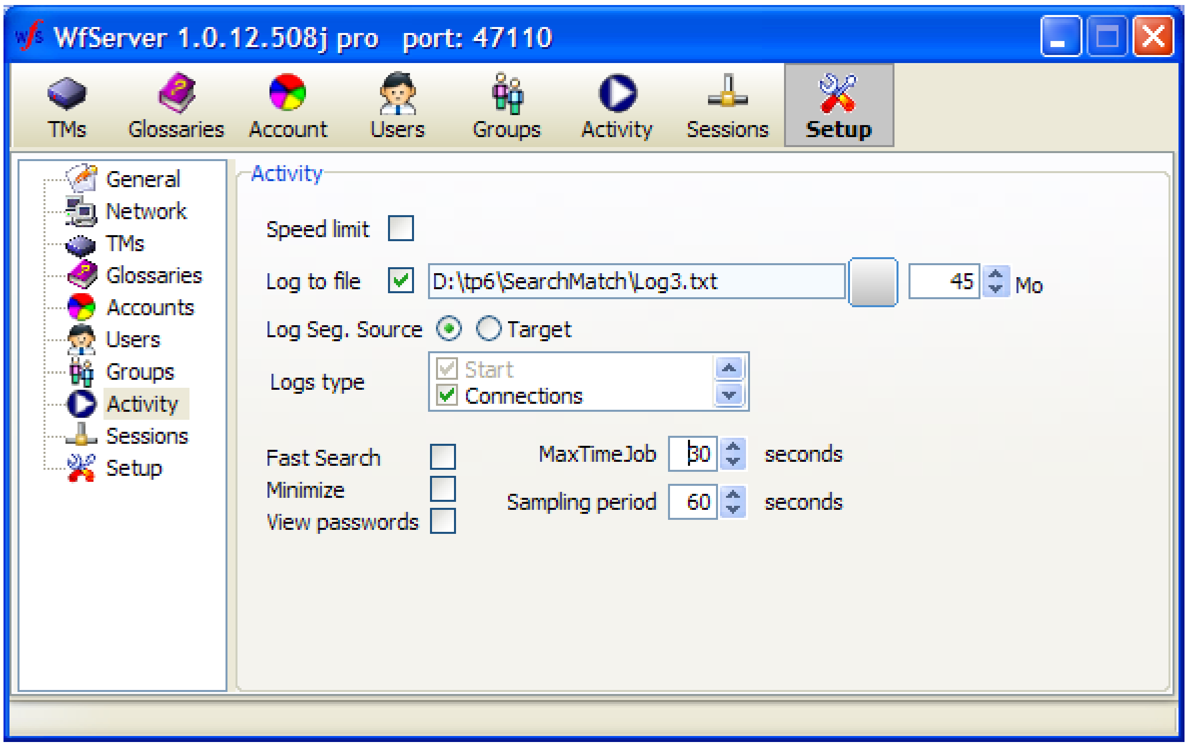
The Admin password can be set here. It is required by any "client" that wishes to connect to the server in Admin (administrator) mode. At this time, no application exists to pilot WFS from a remote location in administration mode, but the API explains how such a client application can connect to WFS in admin mode.

We recommend accessing WFS in *remote desktop* mode to set it up or administer it by directly manipulating its user interface.

The *Maxi* value is the maximum number of persons that can simultaneously access the server in Admin mode.

The Donor password is "Donor" by default. It allows clients accessing the server in "Donor" mode to write public TUs into a TM. "Public" TUs are visible to anyone accessing the server, so they are considered as being "donated" to the public.

Activity



Speed limit ☐

Log to file ✓ D:\tp6\SearchMatch\Log3.txt 45 Mo

Log Seg. Source [✓ Source] Target

Logs type: ✓ Start, ✓ Connections

Fast Search ☐

Minimize ☐

View passwords ☐

MaxTimeJob 30 seconds

Sampling period 60 seconds

Speed limit: If checked, WFS' overall speed is limited to a certain number of requests per **second**.

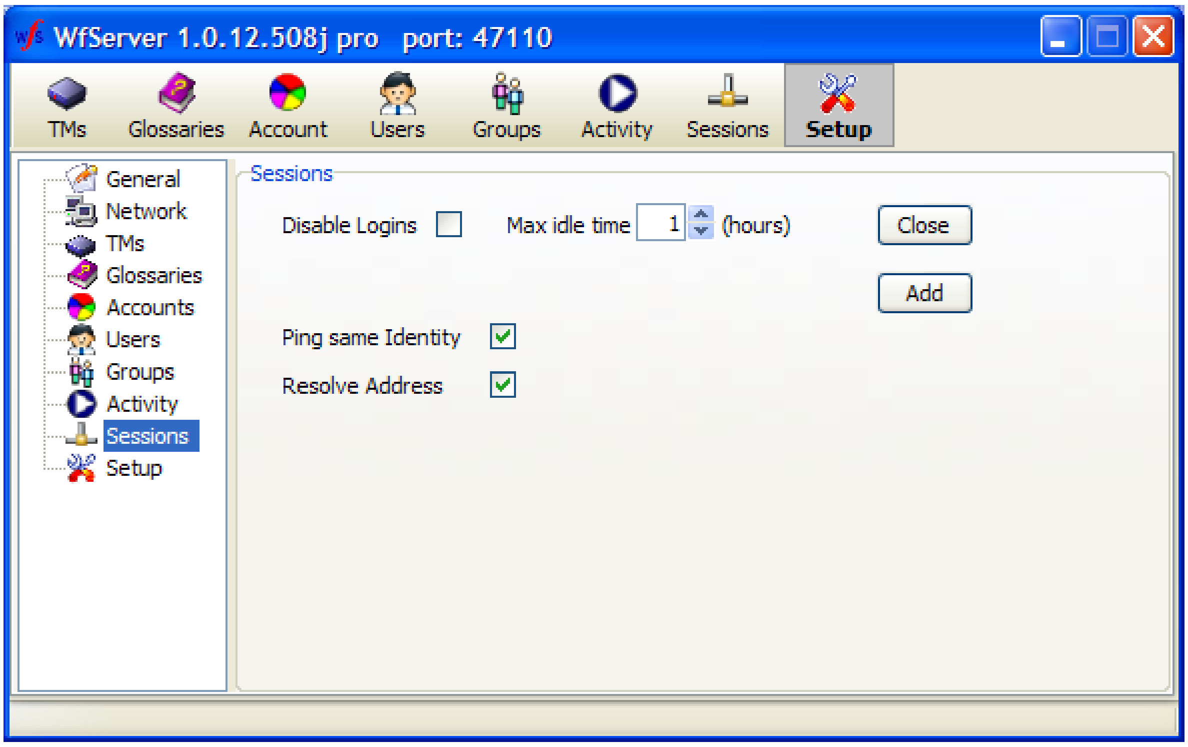
Log to file: This setting lets the administrator specify whether the activity log is written to a log file on disk.

Minimize: specifies whether WFS minimizes itself in the system tray after a while.

MaxTimeJob: This threshold limits the time spent on typical production short jobs, such as search, concordance, TU writes, etc. 30 seconds is the recommended level.

SamplingPeriod: this is the interval at which the graph in the Activity pane is refreshed. The recommended period is 60 seconds to minimize overhead on WFS.

Sessions



Sessions

Disable Logins ☐ Max idle time 1 (hours)

Ping same Identity ✓

Resolve Address ✓

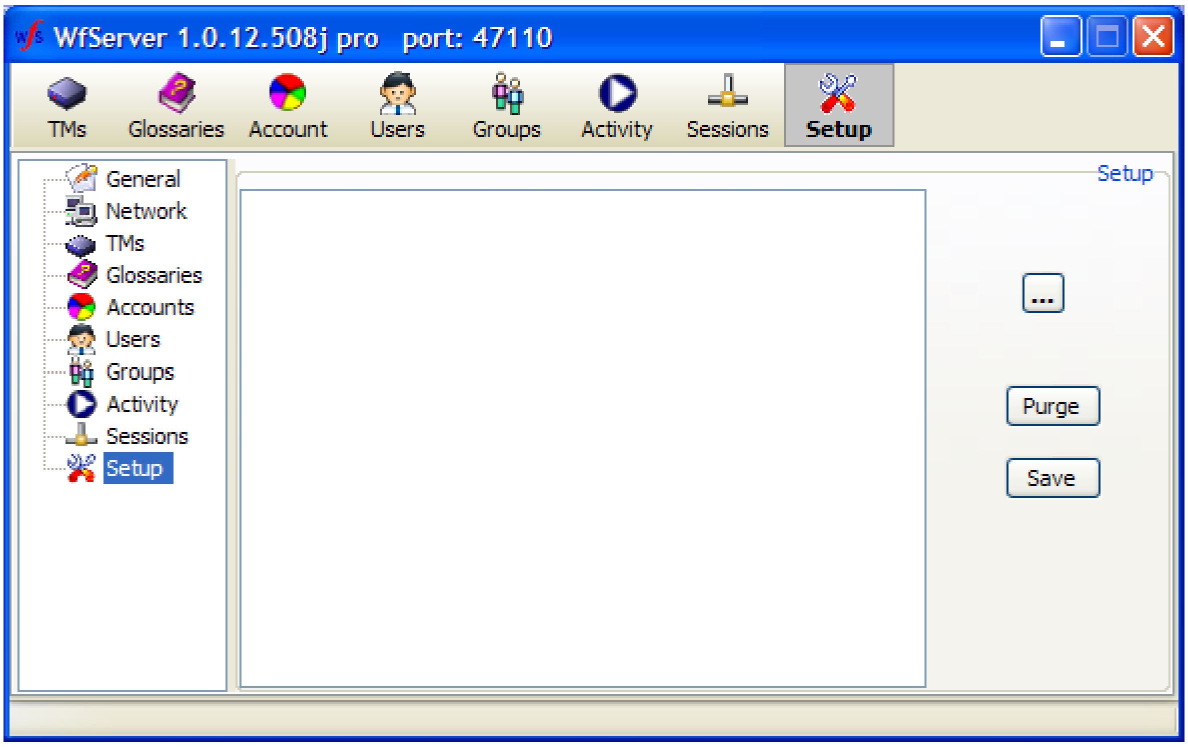
Disable logins: if you want to shut down the server, you may perhaps want to first block any new session from being opened. This checkbox, if checked, rejects any new login request, but keeps existing users logged on until they leave.

Max idle time: this is the maximum idle time until WFS terminates a user's session (the maximum time without a ping or a request coming from a user).

Ping same identity: if two clients are logged with the same identity, this may indicate that one session crashed (or did not end with a proper end-of-session command), and the same user started a new session, with the first still opened. The server will ping both sessions to resolve the problem and know which of the two sessions is "orphaned" and should be ended. Each session has a unique session identifier that makes this operation possible.

Resolve address: tries to get a clear domain name from a user's DNS information.

Setup


WfServer 1.0.12.508j pro port: 47110
TMs
Glossaries
Account
Users
Groups
Activity
Sessions
Setup
General
Network
TMs
Glossaries
Accounts
Users
Groups
Activity
Sessions
Setup
Setup
...
Purge
Save

Appendix 1: Setting up Wordfast Classic or PRO with WFS

Run the following checklist if your client application (like Wordfast Pro, Wordfast Classic, Wordfast Anywhere) cannot connect to WFS.
SomePort usually means 47110, which is Wordfat Server's default port. If the WFS administrator sets up another port, use that port number.
IPnumber is the server's domain name or IP number. If client and server are in the same LAN, use the server's *machine name*, or LAN IP (visible in WFS in the Setup/Network pane). If not working inside a LAN, use the server's WAN IP. The server's WAN IP can be obtained from the server's workstation by pointing a browser at http://www.whatismyip.com. Note that is is preferable to use a *domain name* (or *machine name* from within a LAN).

Methodical two-step troubleshooting:

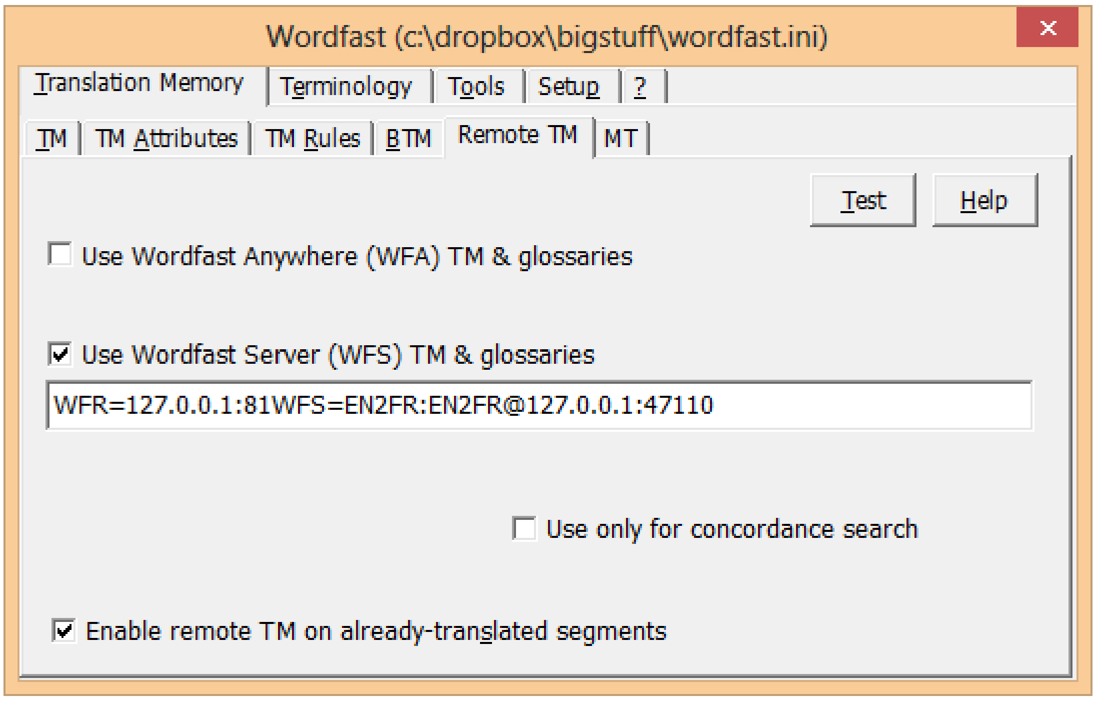
1. Set up WFS in the physical server. Install a client (WFPRO, or WF Classic) in the physical server, and make sure everything works.
2. Set up a client (WF PRO/Classic) in a LAN computer, and test.
If you need to reach WFS from outside the LAN, proceed to set up a client (WF PRO/Classic) in a WAN computer *outside the LAN*, and test.

If 1. works and 2. (LAN or WAN) does not work, you're dealing with port issues, NAT issues, firewall issues, or generally speaking, architecture issues that are outside our hotline's reach. If 1. does not work, see the troubleshooting guidelines below.

With Wordfast **Classic** (WFC):
Prerequisite: you must run a small utility named WfServerRelay.exe alongside WF Server. WF Classic (WFC) actually "talks" to WfServerRelay (WFR). WFR "locally" talks to Wordfast Server (WFS). The physical server should have a fixed IP, and the appropriate port open. WfServerRelay uses port 81 by default.

- ✓ In Wordfast **Classic**, the connection string is entered in the "Remote TM" pane under "Use WFS Wordfast Server (WFS)".
- ✓ Example (using a completely <u>local</u> setting with WfServerRelay + WFS + WFC in the same computer as shown in the screenshots below):
  `WFR=127.0.0.1:81WFS=EN2FR:EN2FR@127.0.0.1:47110`
- ✓ Example (using a <u>distant</u> server; note that only WFR has the distant IP; WFS is seen as "local" by WFR):
  `WFR=227.287.33.45:81WFS=EN2FR:EN2FR@127.0.0.1:47110`

Here are two screenshots that illustrate Wordfast Classic (WFC) running with Wordfast Server (WFS) and Wordfast Relay (WFR) all on the same machine. WFS and/or WFR can be on other machines in which case their respective IP numbers must be updated.



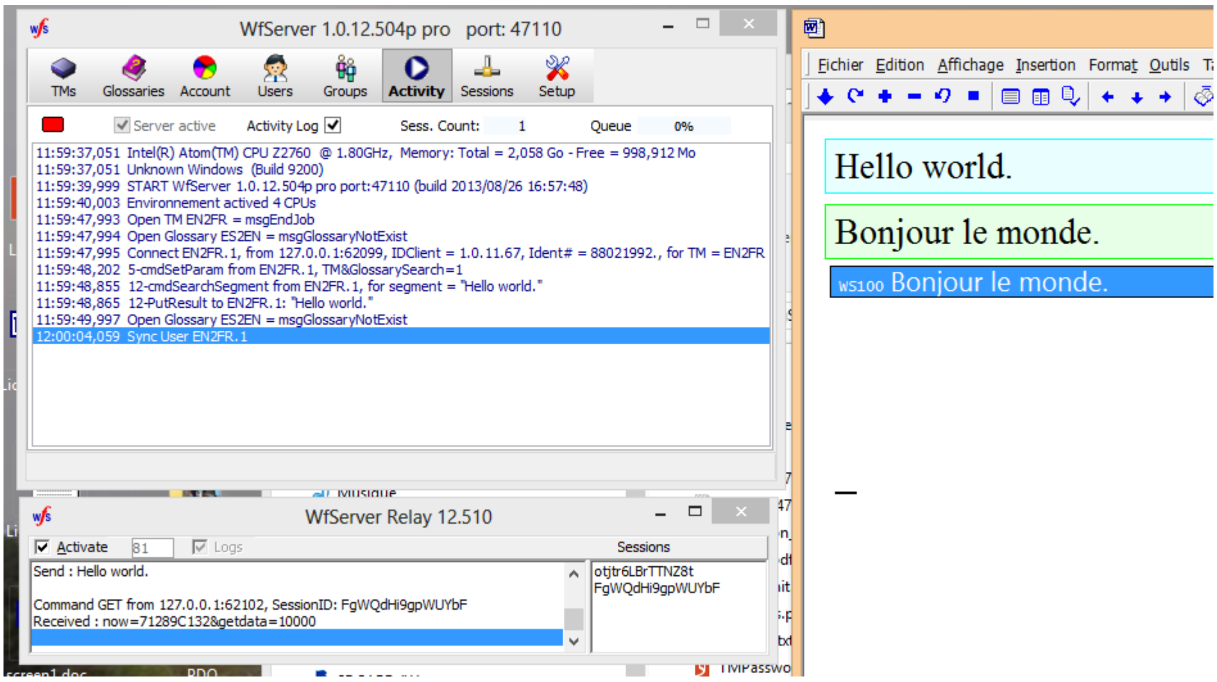
☐ Use Wordfast Anywhere (WFA) TM & glossaries

☑ Use Wordfast Server (WFS) TM & glossaries

WFR=127.0.0.1:81WFS=EN2FR:EN2FR@127.0.0.1:47110

☐ Use only for concordance search

☑ Enable remote TM on already-translated segments



☑ Server active   Activity Log ☑   Sess. Count: 1   Queue 0%

```
11:59:37,051  Intel(R) Atom(TM) CPU Z2760  @ 1.80GHz,  Memory: Total = 2,058 Go - Free = 998,912 Mo
11:59:37,051  Unknown Windows  (Build 9200)
11:59:39,999  START WfServer 1.0.12.504p pro port:47110 (build 2013/08/26 16:57:48)
11:59:40,003  Environnement actived 4 CPUs
11:59:47,993  Open TM EN2FR = msgEndJob
11:59:47,994  Open Glossary ES2EN = msgGlossaryNotExist
11:59:47,995  Connect EN2FR.1, from 127.0.0.1:62099, IDClient = 1.0.11.67, Ident# = 88021992., for TM = EN2FR
11:59:48,202  5-cmdSetParam from EN2FR.1, TM&GlossarySearch=1
11:59:48,855  12-cmdSearchSegment from EN2FR.1, for segment = "Hello world."
11:59:48,865  12-PutResult to EN2FR.1: "Hello world."
11:59:49,997  Open Glossary ES2EN = msgGlossaryNotExist
12:00:04,059  Sync User EN2FR.1
```

WfServer Relay 12.510

☑ Activate 81   ☑ Logs   Sessions

```
Send : Hello world.

Command GET from 127.0.0.1:62102, SessionID: FgWQdHi9gpWUYbF
Received : now=71289C132&getdata=10000
```

```
otjtr6LBrTTNZ8t
FgWQdHi9gpWUYbF
```

Hello world.

Bonjour le monde.

WS100 Bonjour le monde.

For the above example to work:

1. The entire setup (WFC's TM ans setup and the TM used by WFS) is EN-US to FR-FR. A common cause of failure is that the local tool (WFC) and the distant server's TM do not have identical language codes.
2. All software is on the same computer (WFC, WFS, and WFR). If the server and the relay are on other machines, simply update the IP numbers. *If WFR (WfServerRelay) is local and WfServer (WFS) is distant, WFR=... should have 127.0.0.1 as IP (local IP), and WFS=... should have the distant IP address. If both WfServerRelay and WfServer are remote, the WFR=... IP should the remote IP, and the WFS=... IP should then be 127.0.0.1 - because then WFR "talks" to a local WFS).*
3. The *Account* at WFServer is named EN2FR (any other name OK), the password is EN2FR (any other password OK), the "Server active" checkbox is checked in WFServer's "Activity" pane, the "Activity log" checkbox is checked, so that the server echoes the received commands as in the screenshot. In the TM section of WfServer, there must be no password (password field empty) - the password is defined at account level.
4. TM is set to "active" in WFS' *TM* pane.
5. WFClassic is set up as per the screenshot.
6. The active TM in WFServer contains a translation unit with "*Hello World.*" as source.
7. The active TM in WFServer is in "sync" (meaning, it has been reorganized once).

When opening a segment in WFClassic, you should see echoes of your activity in both WfServerRelay and WfServer, similar to what is seen in the screenshots. A "WFS100" match should be served on a green background by WF Classic.

With Wordfast **Pro**, Wordfast **Anywhere**:

- ✓ The correct connection string is entered in the "TM" pane under "Translation Memory".
  Example: wfs://MyLogin:MyPassword@IPnumber:SomePort
  where *MyLogin* is an active WfServer account name.
- ✓ Make sure the language codes of the TM on the server side and those of Wordfast PRO's current project are *identical*. Those language codes are visible as TM properties in the Sessions/TM pane (the TM must be in use), or when opening the actual TM text file with Notepad: read the TM's very first line

With **both** clients:

- ✓ A "real" Wordfast TM has been added to the "TM" pane.
- ✓ An account has been created. The account sets *MyLogin* and *MyPassword* as provided in the connection examples above. The account has a **positive**

number in the small "Nb" setting (the number of clients that are allowed to connect. Do not leave that setting empty).

- ✓ The "Server Active" checkbox in the "Activity" pane is checked. We recommend checking the "Activity log" checkbox to monitor activity when setting up, and unchecking during regular activity.
- ✓ The client's and server's firewalls allows communication through the chosen port (47110 by default).

Appendix 2: Using Wordfast Server with a REST API

WFS can also be launched together with WfServerRelay (also provided in the application installation download, WfServer.zip). In that mode, client applications can communicate with WfServerRelay, who in turn communicates with WFS. WfServerRelay uses HTTP for communication, and responds to REST-like commands, making integration a lot easier. This way, WfServer can be integrated with an existing application in virtually an hour.

Here are the main REST methods to be used with WFS. Query methods are URL-based. The replies from WfServerRelay are HTTP (XMLHTTP) bare text frames that are made of Unicode UTF-16 (16-bit characters), but which do not include an HTML skeleton.

For the example below, we assume that WFS was set up as for the Wordfast Classic example in Annex 1, where *address* means a valid IP number, or a valid domain like http://www.MyWebSite.com. The tests below use a local 127.0.0.1 address + default 81 port for WfServerRelay, 127.0.0.1 + default port 47110 for Wordfast Server, an account named *EN2FR* with a password as *password*. The WFS *EN2FR* account points uses a TM that contains leverage (various translations) for our test sentence.

Server reply format
All found Translation Units (TUs) are in tab-delimited text format, meaning all TUs are separated with ASCII 13 (carriage return), and within each TU, fields are separated with ASCII 9 (tabulators). After the block of TUs, there is an ASCII 16 character followed by all found terminology entries, in the same tab-delimited text format. The TU and glossary format is that of Wordfast Classic TM and glossaries, explained in the Wordfast Classic technical reference manual (www.wordfast.net > Products > Download > technical reference manual or click here).

Here is a full example, where the source segment to query is "*Nice day for a Wordfast Server test.*", and where the reply has one 100% match, one fuzzy 91% match, and two glossary entries, one for *day*, and one for *Server*:

Query (opening a session)

```
http://127.0.0.1:81/?now=17FB794&cmd=22&para=EN2FR%3Apassword%40127%2E0%2E0
%2E1%3A47110
```

Note: this very first command in a session must send an HTTP header where the header's name is "From" and the headers' value is a unique ID that identifies the client application. Keep the ID short, for example, 5 to 15 ASCII characters. The ID should remain the same for one client for the duration of a translation session. It is used to manage accidental deconnection/reconnection during a session, or when multiple clients are connected.

Reply from server:

```
[1.22.7.0.]
```

where 1 means "no error", 22 is an echo of the command, the two other parameters are not used.

The following command instructs WfServer to use the glossary that is attached to the current account. It can be skipped if terminology is not used:

```
http://127.0.0.1:81/?now=23CD794&cmd=49&para=TM%26GlossarySearch%3D1
```

There is no reply to the above command.

Notes:
the `now=token`... parameter is used to sidestep the URL caches on most system. Use it if you needed. Most personal computers cache server replies.
All parameter values are URL-encoded. for example, `%40127%2E0%2E0%2E1%3A47110` means `@127.0.0.1:47110`

Query TM and terminology for a source segment
Now let us query the following source segment: *`Nice day for a Wordfast Server test.`*

```
http://127.0.0.1:81/?now=10C262&cmd=5&para=Nice+day+for+a+Wordfast+Server+t
est%2E
```

Next, we let the server know we expect a reply buffer:

```
http://127.0.0.1:81/?now=20C8454&getdata=20000
```

Note that you may have to introduce a slight delay after this command, depending on how fast the server replies, or check the ReadyState status of your XMLHTTP object . The `&getdata` value (20000) is the max size of the reply buffer in bytes.

Reply from Wf Server

```
10020140106~161446	YC	3	EN-US	Nice day for a Wordfast Server test.	FR-FR	Belle journée pour un essai de Wordfast Server.
09120140106~161446	YC	1	EN-US	Nice day for a Wordfast Server evaluation.	FR-FR	Belle journée pour une évaluation de Wordfast Server.
□
```

day	journée yves' glossary
Server	Serveur yves' glossary

Notes:
The character is ASCII/ANSI 16. It separates the first block of Translation Units (TUs) from the second block, which are glossary entries.
The first three digits in TUs is the match score. The second TU has a match score of 91. Then comes the TU's date+time stamp, translator's initials, re-use counter, source language code, source sentence, target language code, target sentence, and optional meta tags for the TU, also known as "attributes". All fields are separated with tabulators (ASCII 9).

Now we close the session
```
http://127.0.0.1:81/?now=93CE864&cmd=25
```

Other common commands:
15: Concordance search. The `&para=` parameter passes the terms to be looked up, separated with a space (meaning OR), or a plus sign (meaning AND). The server sends back a block of TUs, with up to the 50 best hits. TUs either contain some/all terms (OR), or all terms (AND). DOS-like wildcards can be used such as ? and *.
TUs are sent back with the most relevant at the top. Concordancing is done on the target language. If the reply is larger than the reply buffer block size, more replies are sent, and must be fetched.
9: Delete a TU based on entire TU. The TU to be deleted must be passed as argument. All TUs where all fields (date+time, source segment, target segment, language codes, attributes, etc) match the passed TU are deleted.
Example:
```
http://127.0.0.1:81/?now=10C262&cmd=9&para=20140106%7E161446%09YC%093%09EN%
2DUS%09Nice+day+for+a+Wordfast+Server+test%2E%09FR%2DFR%09Belle+journ%C3%A9
e+pour+un+essai+de+Wordfast+Server%2E%09
```
10: Add a TU. The entire TU must be passed as parameter.
25: Close a translation session.

The full list of commands is in Appendix 3.

Appendix 3: WfServer API commands

List of commands

For most commands, the returned status contains the error message. If the operation ends well, the status contains ***msgEndJob***. If the request continues after a time greater than "***MaxTimeJob***" ("***Setup/Activity***" tab), the answer is "000" with the appropriate error message.

cmdPackIndex = 1

Defragments the index (re-indexes, or reorganizes the TM or glossary). ~ Restricted to Administrators

Details: This command applies to the index of the TM opened during the session. The index is defragmented. During this period, the TM is not accessible. WfServer sends back "000" to users' queries.

cmdCreateTM = 2

Creates a remote TM and indexes the TM. ~ Restricted to Administrators

Parameters: *Parameters are separated with a Tab character (# 9)*

1: Name of the TM
2: Relative path name + TM file.
3: Password for the TM
4: Languages of TM in the format "XX-XX> XX-XX
5: Write permission on the TM ("1" or "0")
6: Write permission on public zone of the TM ("1" or "0")

Details: WfServer creates a TM. This TM is automatically indexed. This command returns msgEndJob but the process runs in the background.

Java code:

```
string Params, Line ;
int ErrorNumber ;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
Line = Connector.WaitResultString();
If (Line.equal("IsAdmin=1")) {
    Params = "TMName" + #9 + "TMFileName.txt" + #9 + "TMPassword" + "EN>FR-CA" + #9 + "1"
    + #9 + "1";
    Connector.PostCommandWaitTicket(Consts.cmdCreateTM, Params);
    ErrorNumber = Connector.Status.Msg;
}
```

cmdIndexTM = 4

Launch a reindexation of a remote TM. ~ Restricted to Administrators

Parameter: Name of the TM

Details: Indexing is performed in the background. During this period, the TM is not accessible. WfServer sends back "000" to users' queries. The error message ***msgTMLocked*** is returned if a user tries to open a session on this TM, lasting all the time of reindexation.

Java code:

```
string Line ;
int ErrorNumber ;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
Line = Connector.WaitResultString();
If (Line.equal("IsAdmin=1")) {
        Connector.PostCommandWaitTicket(Consts.cmdIndexTM, "TMName");
        ErrorNumber = Connector.Status.Msg;
    }
```

cmdSearchSegment = 5

Searches a TU for a given segment. ~ No restriction.
Parameter: Searched segment

Details: WfServer searches and sends back the found TUs, sorted in the order of resemblance, with a 3-digit score at the beginning of each TU. The reuse counter is then incremented if the "Auto increment 100%" is checked ("***Setup/TMs***" tab).

Java code:

```
    String Line;
    TMServerTCPConnector Connector = new TMServerTCPConnector();
    Connector.PostCommand(cmdSearchSegment, "hello world");
    If Connector.WaitLastTicket().Msg = msgEndJob {
        Line = Connector.WaitResultString();
    }
```

cmdDelSegment = 6

Removes one TU for a given segment, if it is found at 100%. ~ No restriction.
Parameter: Searched segment

Details: WfServer searches a TU whose source segment is identical to the searched segment. If a TU is found, it is deleted.

Java code:

```
    TStatus Status = new TStatus();
    TMServerTCPConnector Connector = new TMServerTCPConnector();
    Connector.PostCommand(cmdDelSegment, "hello world");
    Status = Connector.WaitLastTicket();
```

cmdAddSegment = 7

Adds one or many TUs created from one or more source segments. ~ No restriction.
Parameter: Source segment [+Rtn+ source segment] ...

Details: WfServer creates a TU with each given source segment(s), and starts a **cmdAddTUs** command. The langID created will be that of the TM, the GroupID will be that of the session, the target segment will be filled with spaces of the size of the source segment.

Java code:

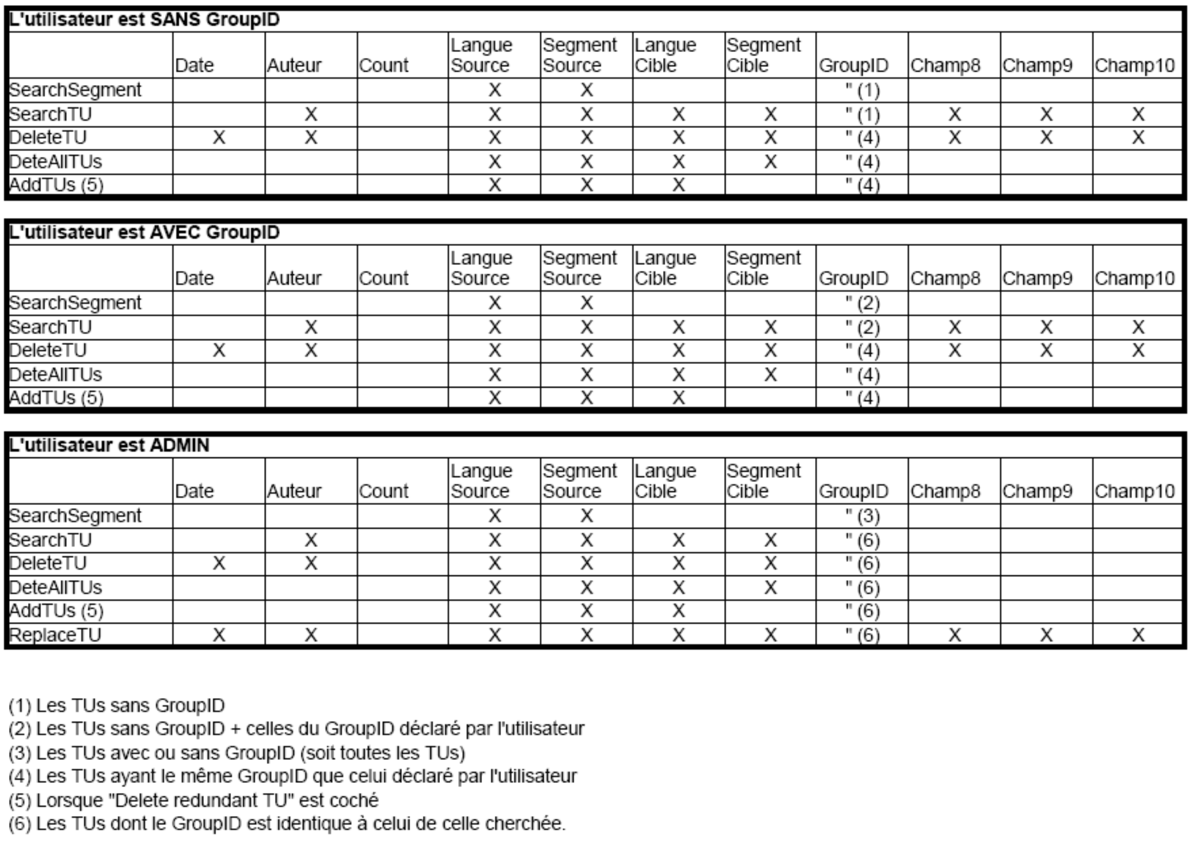
```
    TStatus Status = new TStatus();
    TMServerTCPConnector Connector = new TMServerTCPConnector();
    Connector.PostCommand(cmdAddSegment, "hello world");
    Status = Connector.WaitLastTicket();
```

cmdSearchTU = 8

Search a TU ~ No restriction.
Parameter: Searched TU

Details: WfServer searches and returns the found TU. The reuse counter is then incremented if the "Auto increment 100%" is checked ("Setup / TMs" tab).

The search compares the following fields.



L'utilisateur est SANS GroupID	Date	Auteur	Count	Langue Source	Segment Source	Langue Cible	Segment Cible	GroupID	Champ8	Champ9	Champ10
SearchSegment				X	X			" (1)			
SearchTU		X		X	X	X	X	" (1)	X	X	X
DeleteTU	X	X		X	X	X	X	" (4)	X	X	X
DeteAllTUs				X	X	X	X	" (4)			
AddTUs (5)				X	X	X		" (4)			

L'utilisateur est AVEC GroupID	Date	Auteur	Count	Langue Source	Segment Source	Langue Cible	Segment Cible	GroupID	Champ8	Champ9	Champ10
SearchSegment				X	X			" (2)			
SearchTU		X		X	X	X	X	" (2)	X	X	X
DeleteTU	X	X		X	X	X	X	" (4)	X	X	X
DeteAllTUs				X	X	X	X	" (4)			
AddTUs (5)				X	X	X		" (4)			

L'utilisateur est ADMIN	Date	Auteur	Count	Langue Source	Segment Source	Langue Cible	Segment Cible	GroupID	Champ8	Champ9	Champ10
SearchSegment				X	X			" (3)			
SearchTU		X		X	X	X	X	" (6)			
DeleteTU	X	X		X	X	X	X	" (6)			
DeteAllTUs				X	X	X	X	" (6)			
AddTUs (5)				X	X	X		" (6)			
ReplaceTU	X	X		X	X	X	X	" (6)	X	X	X

(1) Les TUs sans GroupID
(2) Les TUs sans GroupID + celles du GroupID déclaré par l'utilisateur
(3) Les TUs avec ou sans GroupID (soit toutes les TUs)
(4) Les TUs ayant le même GroupID que celui déclaré par l'utilisateur
(5) Lorsque "Delete redundant TU" est coché
(6) Les TUs dont le GroupID est identique à celui de celle cherchée.

Java code:

```
String Line;
String TU;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(cmdSearchTU, TU);
. . . .
If Connector.WaitLastTicket().Msg = msgEndJob {
Line = Connector.WaitResultString();
}
```

cmdDeleteTUs = 9

Deletes a list of TUs. ~ No restriction.

Parameters:

1: TU_1 + Rtn + TU_2 + Rtn + TU_3 + ...

1: 23456 + Rtn + 54678 + Rtn + ...

Details: WfServer searches each transmitted TU and removes the one(s) found as identical (see the field comparison table). On WfServer only one identical TU is removed while WFServer removes all identical TUs that were found. The search can be done in two ways, either by sending WfServer the searched TU, or by sending its identification number recovered during a previous Search command.

Java code:

```
TStatus Status = new TStatus();
String TUs;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(cmdDeleteTUs, TUs);
Status = Connector.WaitLastTicket();
```

cmdAddTUs = 10

Adds a list of TUs. ~ No restriction.
Parameters: TU1 + Rtn + TU2 + Rtn + TU3 + ...

Details: WfServer adds each TU to the session's TM. Before the operation, redundant TUs beings can be automatically deleted if the "***Delete redundant TU***" checkbox ("**Setup/TMs**" tab) is checked. The table below summarizes what is removed taking into account the parameter "***OverrideTU***" (see below "***cmdSetParam***")

Java code:

```
TStatus Status = new TStatus();
String TUs;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(cmdAddTUs, TUs);
Status = Connector.WaitLastTicket();
```

cmdDelLastTU = 11

Removes an older TU **and** adds a new TU, in one operation. ~ No restriction.
Parameters:

1: Identifier of the TU to be deleted (e.g. "45678").

2: TU to be added.

Details: This command allows to remove a TU and add another at the same time. The first TU is referenced by its ID

Java code:

```
TStatus Status = new TStatus();
String TU;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(cmdDelLastTU, "45678" + #13 + TU);
Status = Connector.WaitLastTicket();
```

cmdIncLastTU = 12

Increments the reuse counter of a TU. ~ No restriction.
Parameters: Position of the TU to be incremented

Details: WfServer increments the reuse counter of the designated TU. If the transmitted position is <= 0, the last TU served at 100% will be be incremented.

Java code:

```
TStatus Status = new TStatus();
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(cmdIncLastTU, "45678");
Status = Connector.WaitLastTicket();
```

cmdSearchContext = 15

Searches the TUs whose source segment contains the largest number of searched terms. ~ No restriction.

Parameters: Keywords to be searched, separated by spaces.

Details: WfServer returns packets of TUs to the user. The size of each packet (in number of text characters - not bytes) is defined by a parameter specified in the request. WfServer fills the packet with as many entire TUs as possible.

Java code:

```
public static String NoResultSt = new String("000");
String Line;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSearchContext, "the");
do {
    Line = Connector.WaitResultString;
} while ((Line != null) && (!Line.contentEquals(NoResultSt)));
```

cmdInitFirstTU = 17

Positions a cursor in the TM file. ~ Restricted to Administrator or Donor

Parameters: Number in ASCII. Example: "12345"

Details: The cursor is used to directly read the contents of the TM, one TU at a time, or block by block.

Java code:

```
string Line ;
int ErrorNumber ;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
Line = Connector.WaitResultString();
If (Line.equal("IsAdmin=1")) {
    Connector.PostCommand(cmdInitFirstTU, "45678");
    ErrorNumber = Connector.Status.Msg;
}
```

cmdGetNextTU = 18

Reads the TU under the current cursor in the direction Start -> End .. ~ Restricted to Administrator or Donor

Parameters: Position in ASCII format. e.g. "12345", or empty

Details: WfServer returns the TU under the cursor and moves the cursor to the next TU. Also returns the cursor. All TUs end with # 13. If reached or exceeded the end of file, returns '000 '.

Java code:

```
public static String NoResultSt = new String("000");
String Line;
int ErrorNumber ;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
Line = Connector.WaitResultString();
If (Line.equal("IsAdmin=1")) {
    Connector.PostCommand(Consts.cmdGetNextTU, "0");
    do {
        Line = Connector.WaitResultString;
    } while ((Line != null) && (!Line.contentEquals(NoResultSt)));
}
```

cmdGetPreviousTU = 19

Reads the TU under the current cursor in the direction End -> Start

Restrictions: Administrator or Donor

Parameters: Position in ASCII format. e.g. "12345", or empty

Details: WfServer returns the TU under the cursor (moving backwards) and positions the cursor at the end of the previous TU. Also returns the cursor. All TUs end with # 13. If reached or exceeded the beginning of file, returns '000 '.

Code Java :

```
public static String NoResultSt = new String("000");
String Line;
int ErrorNumber ;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
Line = Connector.WaitResultString();
If (Line.equal("IsAdmin=1")) {
    Connector.PostCommand(Consts.cmdGetPreviousTU, MAXINT);
    do {
        Line = Connector.WaitResultString;
    } while ((Line != null) && (!Line.contentEquals(NoResultSt)));
}
```

cmdGetCountTUs = 20

Returns the number of TUs in the current TM. ~ No restriction.

Parameters: None.

Details: WfServer returns the file name of the TM and the current number of TUs that it contains.

Java code:

```
TStatus Status = new TStatus();
String Line;
Int ResultNum;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Status = Connector.PostCommandWaitStatus(cmdGetCountTUs, null);
Line = Connector.getReplyData(ResultNum);
```

cmdGetTMFileName = 21

Returns the name and size (in characters - not bytes) of the TM file. ~ No restriction.
Parameters: None.

Details: WfServer returns the file name of the TM and the number of characters that it contains.

Java code:

```
TStatus Status = new TStatus();
String FileName;
Int FileSize;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Status = Connector.PostCommandWaitStatus(cmdTMFileName, null);
FileName = Connector.getReplyData(FileSize);
```

cmdGetNextBlock = 23

Reads the TU block under the current cursor in the direction Start -> End .. ~ Restricted to Administrators
Parameters: Position in ASCII format. e.g. "12345", or empty

Details: WfServer returns the block of TUs under the cursor and positions the cursor at the end of the block. Also returns the cursor. All TUs end with # 13. If reached or exceeded the end of file, returns '000 '.

Code Java :

```
public static String NoResultSt = new String("000");
String Line;
int ErrorNumber ;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
Line = Connector.WaitResultString();
If (Line.equal("IsAdmin=1")) {
    Connector.PostCommand(Consts.cmdGetNextBlock, "0");
    do {
        Line = Connector. WaitResultString(32000);
    } while ((Line != null) && (!Line.contentEquals(NoResultSt)));
}
```

cmdGetPreviousBlock = 24

Reads the TU block under the current cursor in the direction End -> Start ~ Restricted to Administrators
Parameters: Position in ASCII format. e.g. "12345", or empty

Details: WfServer returns the block of TUs under the cursor (backward direction) and positions the cursor at the beginning of the block. Also returns the cursor. All TUs end with # 13. If reached or exceeded the beginning of file, returns '000 '.

Java code:

```
public static String NoResultSt = new String("000");
String Line;
int ErrorNumber ;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
Line = Connector.WaitResultString();
```

```
If (Line.equal("IsAdmin=1")) {
    Connector.PostCommand(Consts.cmdGetPreviousBlock, MAXINT);
    do {
        Line = Connector. WaitResultString(32000);
    } while ((Line != null) && (!Line.contentEquals(NoResultSt)));
}
```

cmdGetBlockASync = 26

Reads the TU block under the current cursor in the direction Beginning -> End ~ Restricted to Administrators

Parameters: Position in ASCII format. e.g. "12345", or empty

Details: WfServer returns the block of TUs under the cursor and positions the cursor at the beginning of the block. Also returns the cursor. All TUs end with # 13. If reached or exceeded the beginning of file, returns '000 '.

Java code:

```
public static String NoResultSt = new String("000");
String Line;
int ErrorNumber ;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
Line = Connector.WaitResultString();
If (Line.equal("IsAdmin=1")) {
    Connector.PostCommand(Consts.cmdGetBlockAsync, "435678");
    do {
        Line = Connector. WaitResultString(32000);
    } while ((Line != null) && (!Line.contentEquals(NoResultSt)));
}
```

cmdGetVersionNumber = 28

Returns the version number of WfServer ~ No restriction.

Parameters: None.

Details: WfServer returns its version number.

Java code:

```
String Line;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdGetVersionNumber, null);
Line = Connector.WaitResultString();
```

cmdDoBreak = 29

Stops all the currently running jobs for a user. ~ No restriction.

Parameters: None.

Details: WfServer properly shuts down the execution of a command and deletes all other commands in the job queue.

Java code:

```
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdDoBreak, null);
```

cmdGetErrorMsg = 30

Returns the string corresponding to the error message. ~ No restriction.

Parameters: Error number in ASCII (e.g. "12")

Details: WfServer returns the meaning of the error message in text format.

Java code:

```
string ErrorStr;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdGetErrorMsg, "54");
ErrorStr = Connector.WaitResultString();
// ErrorStr = « Invalid command »
```

cmdCreateGlos = 33

Remotely creates a new glossary. ~ Restricted to Administrators

Parameters:

Each parameter is separated from the previous one with a Tab character (# 9)

1: Name of the glossary
2: Name of file relative to the default directory for glossaries as defined in the "***Setup/Glossaries***" tab.
3: Glossary password.
4: Glossary LangID
5: Write authorization ("1" = yes, "0" = no)
6: Write authorization for the public zone ("1" = yes, "0" = no)

Details: WfServer creates a glossary and automatically indexes it. This command returns ***msgEndJob***, but the process runs in the background.

Java code:

```
string Params, Line ;
int ErrorNumber ;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
Line = Connector.WaitResultString();
If (Line.equal("IsAdmin=1")) {
    Params = "GloName" + #9 + "GloFileName.txt" + #9 + "GloPassword" + "EN>FR-CA" + #9 +
    "1" + #9 + "1";
    Connector.PostCommandWaitTicket(Consts.cmdCreateGlos, Params);
    ErrorNumber = Connector.Status.Msg;
}
```

cmdDeleteGlos = 34

Remotely deletes a glossary. ~ Restricted to Administrators

Parameters: *Parameters are separated with a Tab character (# 9)*

1: Name of the glossary
2: Glossary password.
3: Delete the files (glossary file and index). ("1" = yes, "0" = no)

Details: WfServer send the user a confirmation string (10 random alphanumeric characters) before executing the command. The distant user will have to return this confirmation string before the maximum time of the execution of a command. If the response time exceeds "***MaxTimeJob***"

("***Setup/Activity***" tab), confirmation is canceled, the glossary is not deleted.

Java code:

```
String Line, Params, ConfirmationStr;
int ErrorNumber ;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
Line = Connector.WaitResultString();
If (Line.equal("IsAdmin=1")) {
    Params = "GloName" + #9 + "GloPassword" + #9 + "1";
    Connector.PostCommand(Consts.cmdDeleteGlos, Params);
    ConfirmationStr = Connector.WaitResultString();
    If (Connector.Status.Msg = msgConfirmation) {
        Connector.PostCommand(Consts.cmdConfirmation, ConfirmationStr);
    }
    ErrorNumber = Connector.Status.Msg;
}
```

cmdGetLanguage = 35

Returns the language pair of the TM ~ No restriction.
Parameters: None.

Details: WfServer returns the language pair of the TM (e.g. 'RU-01>EN-US')

Java code:

```
String LangStr;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdGetLanguage, null);
Line = Connector.WaitResultString();
```

cmdDeleteTM = 36

Remotely deletes a TM. ~ Restricted to Administrators
Parameters: None

Each parameter is separated from the previous one with a Tab character (# 9)
1: Name of the TM
Password for the TM
3: Delete the files (TM file and index). ("1" = yes, "0" = no)

Details: WfServer send the user a confirmation string (10 random alphanumeric characters) before executing the command. The distant user will have to return this confirmation string before the maximum time of the execution of a command. If the response time exceeds "***MaxTimeJob***" ("***Setup/Activity***" tab), confirmation is canceled, the TM is not deleted.

Java code:

```
String Line, Params, ConfirmationStr;
int ErrorNumber ;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
Line = Connector.WaitResultString();
If (Line.equal("IsAdmin=1")) {
    Params = "TMName" + #9 + "TMPassword" + #9 + "1";
    Connector.PostCommand(Consts.cmdDeleteTM, Params);
```

```
        ConfirmationStr = Connector.WaitResultString();
        If (Connector.Status.Msg = msgConfirmation) {
            Connector.PostCommand(Consts.cmdConfirmation, ConfirmationStr);
        }
        ErrorNumber = Connector.Status.Msg;
    }
```

cmdCreateAccount = 37

Remotely creates a new account. ~ Restricted to Administrators

Parameters: *Parameters are separated with a Tab character (# 9)*

1: Account Name
2: Password of the account.
3: Name of the associated TM.
4: Write authorization ("1" = yes, "0" = no)
5: Name of the associated glossary.
6: Write authorization for the glossary ("1" = yes, "0" = no)
7: Max number of concurrent users

Details: WfServer remotely creates an account. If the TM does not exist, or the syntax of the name is not correct, or the account already exists, the account is not created.

Java code:

```
    String Line, Params;
    int ErrorNumber ;
    TMServerTCPConnector Connector = new TMServerTCPConnector();
    Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
    Line = Connector.WaitResultString();
    If (Line.equal("IsAdmin=1")) {
         Params = "AccountName" + #9 + "AccountPassword" + #9 + "TMName" + #9 + "1" + #9 +
         "GloName" + #9 + "1" + #9 + "100";
        Connector.PostCommandWaitTicket(Consts.cmdCreateAccount, Params);
        ErrorNumber = Connector.Status.Msg;
    }
```

cmdGetAccntParams = 38

Returns the settings for an account ~ Restricted to Administrators
Parameter: Account Name

Details: WfServer returns:

1: Password of the account.
2: Name of the associated TM.
3: Write authorization ("1" = yes, "0" = no)
4: The name of the glossary
5: Write authorization for the glossary ("1" = yes, "0" = no)

Java code:

```
    String Line;
    int ErrorNumber ;
    TMServerTCPConnector Connector = new TMServerTCPConnector();
    Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
    Line = Connector.WaitResultString();
    If (Line.equal("IsAdmin=1")) {
        Connector.PostCommand(Consts.cmdGetAccntParam, "AccountName");
        Line = Connector.WaitResultString();
        ErrorNumber = Connector.Status.Msg;
    }
```

cmdGetNbClient = 39

Returns the number of clients who opened a session. ~ No restriction.
Parameters: None.

Details: WfServer returns the number (in ASCII) of logged users (real and virtual).
Java code:

```
    String CountStr;
    TMServerTCPConnector Connector = new TMServerTCPConnector();
    Connector.PostCommand(Consts.cmdGetNbClient, null);
    CountStr = Connector.WaitResultString();
```

cmdDeleteAccount = 40

Remotely deletes an account. ~ Restricted to Administrators

Parameters: *Parameters are separated with a Tab character (# 9)*
1: Account Name
2: Password of the account.

Details: WfServer sends the user a confirmation string (10 random alphanumeric characters) before executing the command. The distant user will have to return this confirmation string before the maximum time of the execution of a command. If the response time exceeds "***MaxTimeJob***" ("***Setup/Activity***" tab), confirmation is canceled, the account is not deleted.
Java code:

```
    String Line, Params, ConfirmationStr;
    int ErrorNumber ;
    TMServerTCPConnector Connector = new TMServerTCPConnector();
    Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
    Line = Connector.WaitResultString();
    If (Line.equal("IsAdmin=1")) {
        Params = "Account1" + #9 + "Account2" + #9 + …
        Connector.PostCommand(Consts.cmdDeleteAccount, Params);
        ConfirmationStr = Connector.WaitResultString();
        If (Connector.Status.Msg = msgConfirmation) {
            Connector.PostCommand(Consts.cmdConfirmation, ConfirmationStr);
        }
        ErrorNumber = Connector.Status.Msg;
    }
```

cmdReplaceTU = 44

Replaces a TU by another in the same place (if possible) without changing the GroupID. ~ Restricted to Administrators
Parameters: Each TU is separated by a **Rtn** character (# 13)

1: Former TU.

2: New TU.

Details: This command allows you to replace a TU by another. If both TUs are the same size, the new one replaces the old one.

TU1,	""	Deletes the TU
TU1,	TU2	Replaces TU1 with TU2 if TU1 exists
TU1,	"123"	Invalid parameter
"",	TU2	Replaces the last found TU with TU2
"",	""	Deletes the last found TU
"123",	TU2	Replaces TU # 123 with TU2
"123",	""	Deletes TU # 123
"123",	"432"	Invalid parameters

Java code:

```
String Line, Params;
int ErrorNumber ;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
Line = Connector.WaitResultString();
If (Line.equal("IsAdmin=1")) {
    Params = TU1 + #13 + TU2;
    Connector.PostCommand(Consts.cmdReplaceTU, Params);
    ErrorNumber = Connector.Status.Msg;
}
```

cmdDelAllTUs = 45

Searches and Deletes all found TUs. TUs must be found as 100% identical, except for the date. ~ Restricted to Administrators

Parameters: TUs are separated with a **Rtn** character (# 13)

1: TU_1,
2: TU_2
3: TU_n

Details: This command deletes all TUs identical to a list of TUs, even different date.

Java code:

```
String Line, Params;
int ErrorNumber ;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
Line = Connector.WaitResultString();
If (Line.equal("IsAdmin=1")) {
    Params = TU1 + #13 + TU2 + #13 + …;
    Connector.PostCommand(Consts.cmdDelAllTUs, Params);
    ErrorNumber = Connector.Status.Msg;
}
```

cmdAddSearch = 47

Searches a segment and add TUs that will be stamped with the client GroupID. ~ No restriction.

Parameters: TUs are separated with a **Rtn** character (# 13)

1: Searched segment
2: TU to be added.

Details: This command is twofold. It adds a TU first and then searches for the segment. WfServer searches and sends back the found TUs, sorted in the order of resemblance, with a 3-digit score at the beginning of each TU. The reuse counter is then incremented if the "***Auto increment 100%***" checkbox is checked ("**Setup/TMs**" tab).

Java code:

```
String Line, TU;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Line = "The hotel" + #13 + TU;
Connector.PostCommand(cmdAddSearch, Line);
If Connector.WaitLastTicket().Msg = msgEndJob {
    Line = Connector.WaitResultString();
}
```

cmdInitGroupID = 48

Create / Open / Enable / Closes a session WorkgroupID. ~ No restriction.

Parameters:

1: "0123456789" + # 13 + life duration + # 13 + PIN number
2: "123456789"
3: "123456789" + #13 + "+"
4: ""

Details: The syntax may vary depending on the function. 1=**Create**, 2=**Open**, 3=**Enable**, 4=**Close**. A WorkgroupID consists of 10 characters in the following alphabet:

“#$%+-0123456789<=>?@ ABCDEFGHJKLMNPQRSTUVWXYZabcdefghjkmnpqrstuvwxyz

Life duration is **0, 1, 3, 6, 12, 24** months. "**0**" means "no time limit".

Originally, the structure of the WorkGroupID contained information for the life duration of that WorkGroup ID, as well as a PIN number in order to filter WorkGroupID. However, client software that identify themselves at the time of connection to WfServer with a version number less than "1.0.11.30" are not filtered.

It is possible to force this filter by checking the "***PIN filter***" checkbox (tab "***Setup / Groups***")

Java code:

```
String Line;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Line = "0123456789" + #13 + durée de vie + #13 + numéro PIN;
Connector.PostCommandWaitTicket(cmdInitGroupID, Line);
If Connector.Status.Msg = msgEndJob {
}
```

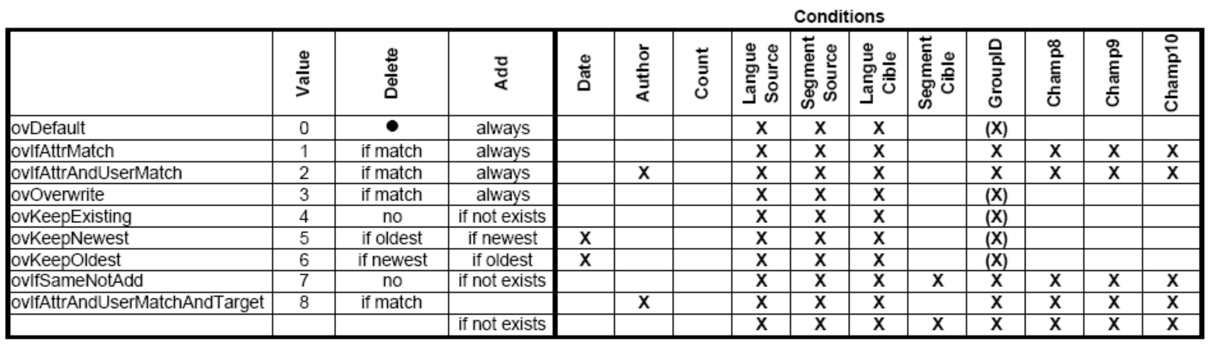
cmdSetParam = 49

Sets one or more global or local parameters. ~ No restriction.

Parameters:

'AnyLangID'	The cmdAppendTM sub-command (cmdPurge=59) accepts all added TUs regardless of language codes ("1"), or rejects TUs that don't have the correct language codes ("0").
'AutoIncTU'	Is the auto-incrementation of TUs served at 100% enabled for my user session?
'AutoIncTUTM'	Is the auto-incrementation of TUs served at 100% enabled for that TM?
'Break'	Stops the job specified by the session number in ASCII.
'CacheIndex'	Enables or disables the index cache. (1 = active)
'CanWritePublicTM'	Provisionally enables or disables the public zone write authorization of the current TM, without changing the corresponding setting in the User Interface. (1 = active)
'CanWriteTM'	Temporarily enables or disables the write authorization for the current TM. Temporarily means that the corresponding setting in the User Interface is not modified. This administrator command blocks the write mode for a TM to all other users accessing it. The parameter stays active as long as the TM remains opened by any other user. It will resume the User Interface's status at the next virgin "Open TM" command. (1 = active)
'CCountUsers'	Corrects the difference between the number of *logged* and *connected* users
'CountAdmins'	Sets the maximum number of concurrent administrators defined on the interface.
'CountBkgJobs'	Set the maximum number of jobs allowed in the background.
'CountErrors'	Resets the error counter to 0.
	Temporarily (session-wide) enables or disables the deletion of duplicate TUs when ***cmdAddTUs*** is used (does not modifiy the User Interface setting).
'DelCopiesEntries'	Temporarily (session-wide) enables or disables the deletion of duplicate TUs when ***cmdAddEntries*** is used (does not modifiy the User Interface setting).
'Donor'	Changes the property of the user into "donor", giving the password "Donor" ("Setup / Users" tab)
'Donor'	Sets the password of the Donor WfServer interface (if Admin).
'ExportPath'	Sets the destination directory for files generated by Purge commands.

'FuzzyThreshold'	Sets the threshold percentage for TUs that are interesting to harvest for a search (ranges from 50% TUs to 95%). For a threshold of 95%, only TUs with a score >= 95% will be returned to the user. The higher the threshold, the lower will be the number of found TUs.
'GloPassword'	The user identifies the glossary password.
'GroupID'	Opens a session WorkGroupID.
'IdleTime'	Sets the maximum number of hours (1 .. 24) of inactivity.
'IndexUsed'	Defines the index used. (Source = 0, 1 = Target)
'IsAdmin'	Changes to administrator mode after the recognition of the administrator password.
'Kill'	Force the closing a TM or a glossary, disconnecting all users that may be currently using this TM or glossary. (Admin)
'LangID'	Defines the session langID.
'LogsFileActived'	Enables (1) or disables (0) the writing into an activity log file for WfServer.
'MakeBackStats'	Creates a STATS.BAK file.
'MaxTimeJob'	Sets the number of seconds for the execution of an application (2 .. 60).
'OkSendMail'	Enables / disables the sending of warning emails.
'OverrideDate'	Authorizes the stamping of the TU date by the server. (Default = 1)
'OverrideTU'	Variable that indicates which TUs will be replaced or removed in a ***cmdAddTUs*** command. The table below summarizes all cases.



				Conditions										
	Value	Delete	Add	Date	Author	Count	Langue Source	Segment Source	Langue Cible	Segment Cible	GroupID	Champ8	Champ9	Champ10
ovDefault	0	●	always				X	X	X		(X)			
ovIfAttrMatch	1	if match	always				X	X	X		X	X	X	X
ovIfAttrAndUserMatch	2	if match	always		X		X	X	X		X	X	X	X
ovOverwrite	3	if match	always				X	X	X		(X)			
ovKeepExisting	4	no	if not exists				X	X	X		(X)			
ovKeepNewest	5	if oldest	if newest	X			X	X	X		(X)			
ovKeepOldest	6	if newest	if oldest	X			X	X	X		(X)			
ovIfSameNotAdd	7	no	if not exists				X	X	X	X	X	X	X	X
ovIfAttrAndUserMatchAndTarget	8	if match			X		X	X	X		X	X	X	X
			if not exists				X	X	X	X	X	X	X	X

● if match and "Delete redundant TU checked"
(X) yes if not Admin

'PackReorganize'	Enables / Disables the option of compacting holes in the TM when reindexing ("***Setup / TMs***" tab).
'PingToAll'	Sends a ping to all users.
'PostStats'	Sends an email alert test with the comment you provide.

'Priority'	Sets the priority level (1 .. 10) for the user without overriding the account level set in the User Interface (in the "***Accounts / Priority Level***" tab)
'SearchDepth'	Sets the level of depth for searches (50 .. 95).
'SetGloPassword'	Sets the glossary password.
'SetTMPassword'	Sets the password for the TM.
'Showserver'	Make the WfServer User Interface visible (1) or invisible (0) .
'SyncTM'	Synchronization (force write) of TM data in RAM to the hard disk.
'SyncUser'	Synchronization (force write) of user data in RAM to the hard disk.
'TM & GlossarySearch'	Enables / Disables the simultaneous search TM + Glossary.
'TMLangID'	Configure the TM's langID pair.
'tmPassword'	Temporarily sets the password of the current TM without overwriting its definition on the interface (***TMs*** tab)
'TurboMode'	Activates the Turbo mode for short sentences research (1, 2, 3 words).
'Unlock'	Unlocks all critical area locks of the designated TM or glossary.
'ViewLogs'	Checks the "***Log to file***" checkbox that generates log files. (***Setup/Activity*** tab)

Details: **Java code**:

```
String Line;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Line = "Priority=10;SearchDepth=60";
Connector.PostCommandWaitTicket(cmdSetParam, Line);
If Connector.Status.Msg = msgEndJob {
}
```

cmdGetParam = 50

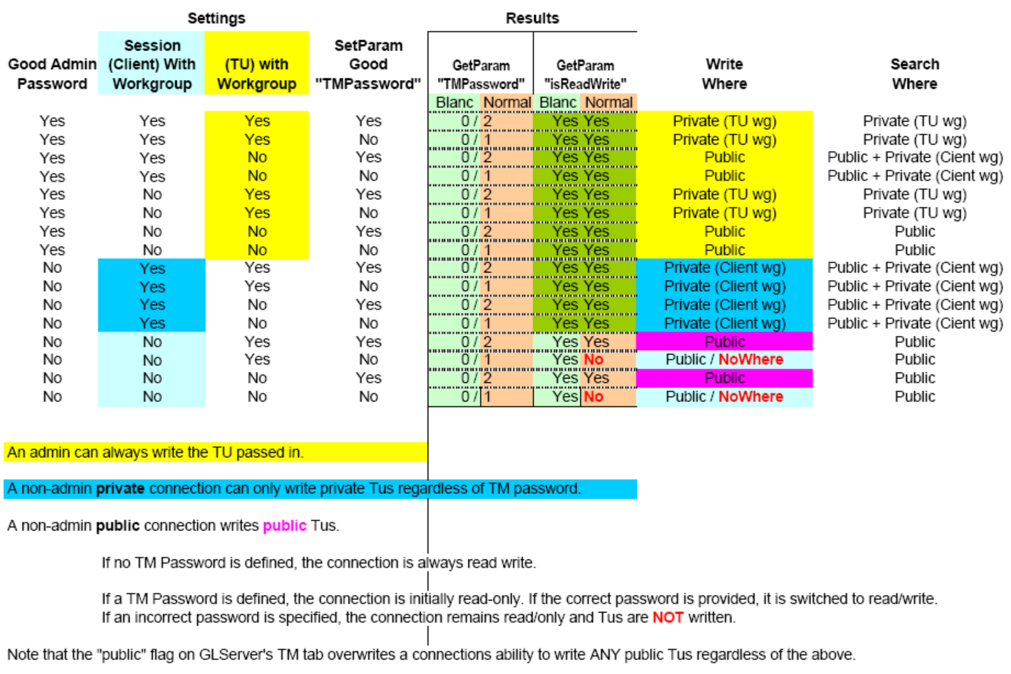
Returns one or many global or local parameters of the session. ~ No restriction.
Parameters:

'AppCPUUsage'	Returns the percentage of CPU activity dedicated to server VLTM.
'AutoIncTU'	Is the auto-incrementation of TUs served at 100% enabled for my user session?
'AutoIncTUTM'	Is the auto-incrementation of TUs served at 100% enabled for that TM?
'CacheIndex'	Returns the state of the index cache. (1 = active)
'CanWritePublicTM'	Retourne the activation status of the write permission into the public area in the current TM. (1 = active)

'CanWriteTM'	Returns the state of the (temporary) write authorization for the current TM. (1 = active)
'CountAccounts'	Returns the number of opened accounts.
'CountActiveThreadsJob'	Returns the number of ThreadJob in progress.
'CountAdmins'	Returns the maximum number of concurrent administrators.
'CountAllTUsID'	Return the number of TUs of all TMs with the WorkgroupID indicated (in Admin mode). If no WorkGroupID is indicated, returns the number of TUs with the same session WorkGroupID, in all TMs.
'CountBkgJobs'	Set the maximum number of jobs allowed in the background.
'CountEntries'	Returns the number of entries in the glossary.
'CountEntriesID'	Returns the number of entries in the current glossary with the same WorkgoupID and LangID as the session user.
'CountErrors'	Returns the number of errors recorded by WfServer.
'CountGlossaries'	Returns the number of opened glossary.
'CountHoles'	Returns the number of TUs in the current TM.
'CountJobs'	Returns the number of Jobs run since the start of WfServer.
'CountReqs'	Returns the number of requests received by the server.
'CountReqsFifo'	Returns the number of requests in the queue.
'CountReqsTM'	Renvoie le nombre de requêtes exécutées sur la TM courante.
'CountSessions'	Returns the number of opened sessions on WfServer.
'CountStart'	Returns the number of times the current version of WfServer was started.
'CountTCPUsers'	Returns the number of connected users.
'CountThreadsJob'	Returns the number of ThreadsJob in progress.
'CountTMs'	Returns the number of opened TMs.
'CountTUs'	Returns the number of TUs in the current TM.
'CountTUsID'	Returns the number of TUs in the current TM with the specified WorkgroupID (in Admin mode). If no WorkGroupID is indicated, returns the number of TUs with the same session WorkGroupID, in the current TM.
'CountTUsLangID'	Returns the number of TUs in the current TM with the same LangID (in Admin mode). If no LangID is indicated, returns the number of TUs with the same session LangID, in the current TM.

'CountUsers'	Returns the total number of logged users.
'CountUsersTM'	Returns the number of users logged into the same TM.
'CPUUsage'	Returns a numeric character string indicating the percentage of CPU usage by WfServer.
'CreatedDate'	Returns the creation date of the WfServer executable.
'DelCopies'	Returns the state of the variable of duplicate TU deletion when ***cmdAddTUs*** is running.
'DelCopiesEntries'	Returns the state of the variable of duplicate entry deletion when ***cmdAddEntries*** is running.
'DiskActivity'	Returns a string of ASCII characters of disk activity expressed in Kb/s
'Donor'	Returns 1 if the user is set as a donor, 0 otherwise.
'ExportPath'	Returns the destination directory for files generated by Purge commands.
'FuzzyThreshold'	Returns the threshold percentage for TUs that are interesting to harvest for a search (ranges from 50% TUs .. 95%). For a threshold of 95%, only TUs with a score >= 95% will be returned to the user. The higher the threshold, the lower will be the number of found TUs.
'GetCountEntry'	Counts the number of specific glossary entries. Sub-parameters, separated by "&": 1 - WorkGroupID ("" = search empty elements, '*' = all items) 2 - Source language (eg 'EN', 'EN*', 'EN-US') 3 - Target language (eg 'EN',' FR*','FR-CA ') 4 - Name of the author or owner.
'GetCountTU'	Count the number of specific TUs in the TM. Sub-parameters, separated by "&": 1 - WorkGroupID ("" = search empty elements, '*' = all items) 2 - Source language (eg 'EN', 'EN*', 'EN-US') 3 - Target language (eg 'EN',' FR*','FR-CA ') 4 - Name of the author or owner.
'GetDateGroupID'	Returns the date contained in the encryption of a WorkgroupID
'GetDateNow'	Returns a string of ASCII characters of the current date (format: ddd:hh:mm:ss,ms), of the operating system on which WfServer is running.
'GetPIN'	Returns the PIN code (4 characters) of a WorkgroupID. If the result is 0, then the WorkgroupID is not syntactically correct with WfServer of versions greater than or equal to "1.0.11.30".

'gloFileName'	Returns the path and file name of the current glossary.
'GloPassword'	Returns the status of the glossary password recognition. 0 = The glossary has a blank password 1 = Password is not identical 2 = The given password is authenticated (identical).
'GlossaryName'	Returns the name of the glossary of the specified account (Admin mode). If none is specified, returns the name of the current glossary.
'GroupID'	Returns the status of the opened WorkgroupID. 0 = not opened, 1 = opened.
'IdleTime'	Returns the maximum number of hours (1 .. 24) of inactivity.
'IndexUsed'	Returns the index used. (Source = 0, 1 = Target)
'IndexVersion'	Returns the version number of the algorithm used to index.
'Indicators'	Returns the progress indicators of the background task specified by its session number.
'IsAdmin'	Returns the administrator status (1 = yes, 0 = no).
'IsDemoMode'	Returns the status of WfServer demo mode (1 = yes, 0 = no)
'IsDoubleIndex'	Returns "1" if target-side searches are possible, otherwise "0".
'IsReadWrite'	Returns the write permission to the TM according to various parameters.



Settings				Results					
Good Admin Password	Session (Client) With Workgroup	(TU) with Workgroup	SetParam Good "TMPassword"	GetParam "TMPassword"		GetParam "isReadWrite"		Write Where	Search Where
				Blanc	Normal	Blanc	Normal		
Yes	Yes	Yes	Yes	0 /	2	Yes	Yes	Private (TU wg)	Private (TU wg)
Yes	Yes	Yes	No	0 /	1	Yes	Yes	Private (TU wg)	Private (TU wg)
Yes	Yes	No	Yes	0 /	2	Yes	Yes	Public	Public + Private (Cient wg)
Yes	Yes	No	No	0 /	1	Yes	Yes	Public	Public + Private (Cient wg)
Yes	No	Yes	Yes	0 /	2	Yes	Yes	Private (TU wg)	Private (TU wg)
Yes	No	Yes	No	0 /	1	Yes	Yes	Private (TU wg)	Private (TU wg)
Yes	No	No	Yes	0 /	2	Yes	Yes	Public	Public
Yes	No	No	No	0 /	1	Yes	Yes	Public	Public
No	Yes	Yes	Yes	0 /	2	Yes	Yes	Private (Client wg)	Public + Private (Cient wg)
No	Yes	Yes	No	0 /	1	Yes	Yes	Private (Client wg)	Public + Private (Cient wg)
No	Yes	No	Yes	0 /	2	Yes	Yes	Private (Client wg)	Public + Private (Cient wg)
No	Yes	No	No	0 /	1	Yes	Yes	Private (Client wg)	Public + Private (Cient wg)
No	No	Yes	Yes	0 /	2	Yes	Yes	Public	Public
No	No	Yes	No	0 /	1	Yes	No	Public / NoWhere	Public
No	No	No	Yes	0 /	2	Yes	Yes	Public	Public
No	No	No	No	0 /	1	Yes	No	Public / NoWhere	Public

An admin can always write the TU passed in.

A non-admin **private** connection can only write private Tus regardless of TM password.

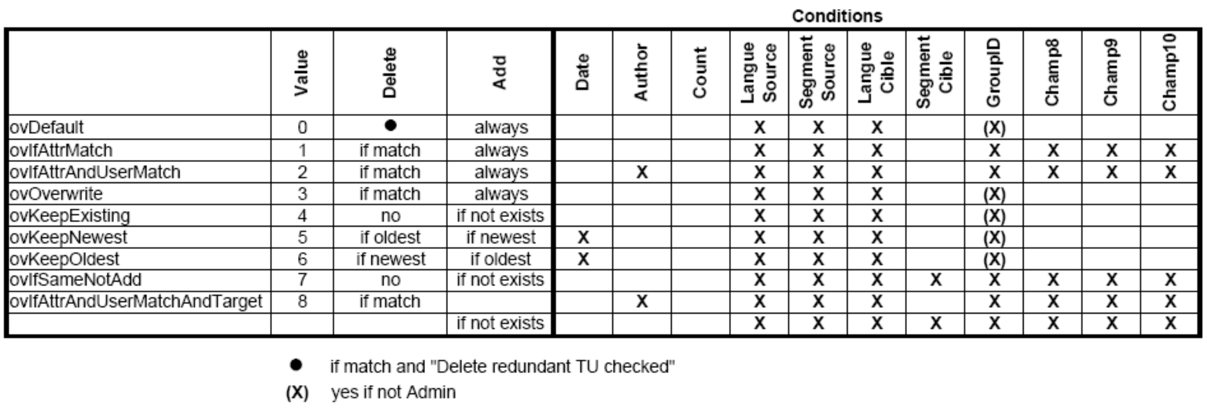
A non-admin **public** connection writes public Tus.

If no TM Password is defined, the connection is always read write.

If a TM Password is defined, the connection is initially read-only. If the correct password is provided, it is switched to read/write.
If an incorrect password is specified, the connection remains read/only and Tus are **NOT** written.

Note that the "public" flag on GLServer's TM tab overwrites a connections ability to write ANY public Tus regardless of the above.

'***IsUnicode***'	Returns the formatting (encoding) of the TM file. (1 = ANSI, 2 = Unicode, 3 = BigEndian, 4 = UTF8).
'***LangID***'	Returns the session langID.
'***Langs***'	Returns the pair of LangID of the TM (e.g. "EN> FR").
'***LiCapacity***'	Returns the maximum number of connections that the license number can accept.
'***LiCount***'	Returns how many connections are used by the license number (on the same operating system installed).
'***ListAllGroupIDs***'	Returns a list, separated by the Tab character (# 9), of all WorkgroupID used in all TMs.
'***ListGroupIDs***'	Returns a list, separated by the Tab character (# 9), of all WorkgroupID used in the current TM.
'***MaxTimeJob***'	Returns the number of seconds for the execution of an application (2 ...60).
'***MemUsed***'	Returns the amount of RAM used, in Kb
'***OkSendMail***'	Returns the checkbox status for sending alert mails ("***Setup/Network***" tab).
'***OverrideDate***'	Returns the status of the authorization to stamp the TUs with the server's date.
'***OverrideTU***'	Returns a variable that indicates which TUs will be replaced or removed in a ***cmdAddTUs*** command.



				Conditions										
	Value	**Delete**	**Add**	**Date**	**Author**	**Count**	**Langue Source**	**Segment Source**	**Langue Cible**	**Segment Cible**	**GroupID**	**Champ8**	**Champ9**	**Champ10**
ovDefault	0	●	always				X	X	X		(X)			
ovIfAttrMatch	1	if match	always				X	X	X		X	X	X	X
ovIfAttrAndUserMatch	2	if match	always		X		X	X	X		X	X	X	X
ovOverwrite	3	if match	always				X	X	X		(X)			
ovKeepExisting	4	no	if not exists				X	X	X		(X)			
ovKeepNewest	5	if oldest	if newest	X			X	X	X		(X)			
ovKeepOldest	6	if newest	if oldest	X			X	X	X		(X)			
ovIfSameNotAdd	7	no	if not exists				X	X	X	X	X	X	X	X
ovIfAttrAndUserMatchAndTarget	8	if match			X		X	X	X		X	X	X	X
			if not exists				X	X	X	X	X	X	X	X

● if match and "Delete redundant TU checked"
(X) yes if not Admin

'***PackReorganize***'	Returns the activation state of the compaction of holes at the time of TM indexing option.
'***Priority***'	Returns the priority level of the user.
'***Showserver***'	Returns 1 if the interface is visible or 0 if the interface is invisible.
'***StatsClients***'	Returns the list of users and their connection status (IP, TM used, connection date)
'***SVersion***'	Returns the version number of WfServer, in ASCII format.

'TM & GlossarySearch'	Returns the activation status of the simultaneous search "TM + Glossary".
'TM_ID'	Returns the ID of the TM.
'TMFileName'	Returns the name of the TM file
'TMFileSize'	Returns the size (in bytes) of the TM.
'TMLangID'	Returns the language pair of the TM.
'TMName'	Returns the name of the TM of the specified account (Admin mode). If none is specified, returns the name of the current TM.
'tmPassword'	Returns the status of the TM password recognition. 0 = The TM has a blank password 1 = Password is not identical 2 = The given password is authenticated (identical).
'TurboMode'	Returns the status of Turbo mode.
'UpTime'	Returns the uptime of WfServer with the format: *yyyy:ddd:hh:mm:ss:ms*.
'UserID'	Returns the ID number of the user session.
'ViewLogs'	Returns the state of the "***Log to file***" checkbox (***Setup/Activity*** tab) box

Details: **Java code**:

```
String Line;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Line = "Priority;SearchDepth";
Connector.PostCommand (cmdGetParam, Line);
If Connector.Status.Msg = msgEndJob {
    Line = Connector.WaitResultString();
}
```

cmdSearchEntry = 51

Searches for an entry in the glossary. ~ No restriction.
Parameters: Source term or expression to search.

Java code:

```
String Line;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(cmdSearchEntry, "hello world");
If Connector.WaitLastTicket().Msg = msgEndJob {
    Line = Connector.WaitResultString();
}
```

cmdAddEntries = 52

Adds one or more entries in the glossary. ~ No restriction.
Parameters: Entries are separated with a **Rtn** character (# 13)
1: Entry$_1$
2: Entry$_2$

Details: The entry is added, the index is updated.
Java code:

```
String Line, Entry1, Entry2;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Line = Entry1 + #13 + Entry2 + #13 + …
Connector.PostCommandWaitTicket(cmdAddEntries, Line);
If Connector.Status.Msg = msgEndJob {
}
```

cmdDelEntry = 53

Deletes an entry in the glossary. ~ No restriction.
Parameters: Entry

Details: The entry is deleted if it exists, the index is updated.
Java code:

```
String Line, Entry;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommandWaitTicket(cmdDelEntry, Entry);
If Connector.Status.Msg = msgEndJob {
}
```

cmdGetNextEntry = 54

Direct reading of the glossary entry by entry in the direction Start -> End. ~ Restricted to Administrators
Parameters: Position in ASCII format. e.g. "12345", or empty.

Details: WfServer returns the entry under the cursor and positions itself on the following entry. Also returns the cursor. All entries end with # 13. If reaches or exceeds the end of file, returns '000 '.
Java code:

```
public static String NoResultSt = new String("000");
String Line;
int ErrorNumber ;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
Line = Connector.WaitResultString();
If (Line.equal("IsAdmin=1")) {
    Connector.PostCommand(Consts.cmdGetNextEntry, "0");
    do {
        Line = Connector.WaitResultString;
    } while ((Line != null) && (!Line.contentEquals(NoResultSt)));
}
```

cmdGetPreviousEntry = 55

Direct reading of the glossary entry by entry in the direction End -> Start. ~ Restricted to Administrators
Parameters: Position in ASCII format. e.g. "12345", or empty.

Details: WfServer returns the entry under the cursor and positions itself on the previous entry. Also returns the cursor. All entries end with # 13. If reaches or exceeds the beginning of file, returns '000 '.
Java code:

```
public static String NoResultSt = new String("000");
String Line;
int ErrorNumber ;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
Line = Connector.WaitResultString();
If (Line.equal("IsAdmin=1")) {
    Connector.PostCommand(Consts.cmdGetPreviousEntry, MAXINT);
    do {
        Line = Connector.WaitResultString;
    } while ((Line != null) && (!Line.contentEquals(NoResultSt)));
}
```

cmdDelEntriesID = 56

Removes all entries from the defined WorkgroupID. ~ Restricted to Administrators
Parameters: WorkgroupID.

Details: The index is updated at each deletion.
Java code:

```
String Line, WorkGroupID;
int ErrorNumber ;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
Line = Connector.WaitResultString();
If (Line.equal("IsAdmin=1")) {
    WorkGroupID = "1234567890";
    Connector.PostCommandWaitTicket(Consts.cmdDelEntryID, WorkGroupID);
    ErrorNumber = Connector.Status.Msg;
}
```

cmdIndexGlos = 58

Remotely launches glossary reindexing. ~ Restricted to Administrators
Parameters: Name of the glossary

Details: Indexing is performed in the background. During this period, the glossary is not accessible. WfServer sends back "000" to users' queries. The error message is ***msgGlossaryLocked***, sent to users during the time of indexing.
Java code:

```
String Line, GlossaryName;
int ErrorNumber ;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
Line = Connector.WaitResultString();
If (Line.equal("IsAdmin=1")) {
    GlossaryName = "EN-US";
    Connector.PostCommandWaitTicket(Consts.cmdIndexGlos, GlossaryName);
    ErrorNumber = Connector.Status.Msg;
}
```

cmdPurgeTM = 59

Execution of a background task on the entire TM. ~ Restricted to Administrators
Parameters:

1. Name of the TM on which the operation will be run, followed by a space " ";

2.1. "**G**" Operation on GroupIDs

- **"D"** Delete
- **"E"** Exports to a file with the name of the TM and the session number

```
59, XX2XX  GDE * EN   FR
59, XX2XX  GDE & EN   FR
59, XX2XX  GDE   EN   FR
59, XX2XX  GDE * EN*  FR*
59, XX2XX  GDE & EN*  FR*
59, XX2XX  GDE   EN*  FR*
59, XX2XX  GDE wkg1  wkg2  wkg3  wkg4  wkg5 ...
```

```
'G' = WorkGroupID
'D' = Delete
'E' = Export
```

```
"*" = all TUs (public and private)
"&" = only private TUs
""  = only public TUs
```

2.2. **"U"** Operation on an individual user

- **"D"** Delete
- **"E"** Exports to a file with the name of the TM and the session number

```
59, XX2XX  UDE wkg1  EN FR #%nom%#
```

2.3. "**A**" Add TM

- **"TMFileName.txt "** file name of the TM to be added

Details: Three possible function, "**G**" = WorkGroupIds, "**U**" = Users, "**A**" = Add a TM. Additional parameters depend on the selected function. **"DE"** for **"G"** or **"U"**, and a file name for **"A".** The server returns a session number so that you can follow its execution in the background.

Code Java :

```
String Line, Progress, Params;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
Line = Connector.WaitResultString();
If (Line.equal("IsAdmin=1")) {
    Params = "TMName" + #9 + "GDE" + #9 + "*" + #9 + "EN" + #9 + "FR";
    Connector.PostCommand(Consts.cmdPurgeTM, Params);
    Line = "Indicators=" + Connector.WaitResultString();
    Do {
        Connector.PostCommand(cmdGetParam, Line);
        Progress = Connector.WaitResultString();
    } While ((Progress.indexof("Msg=0") >= 0);
}
```

cmdPurgeGlo = 60

Running a background task throughout the glossary. ~ Restricted to Administrators
Parameters:

1. Name of the glossary on which the operation will be run, followed by a space " ";

2.1. "**G**" Operation on WorkGroupIDs

- **"D"** Delete
- **"E"** Exports to a file with the name of the TM and the session number

```
60, XX2XX  GDE * EN   FR
60, XX2XX  GDE & EN   FR
60, XX2XX  GDE   EN   FR
60, XX2XX  GDE * EN*  FR*
60, XX2XX  GDE & EN*  FR*
60, XX2XX  GDE   EN*  FR*
60, XX2XX  GDE wkg1  wkg2  wkg3  wkg4  wkg5 ...

'G' = WorkGroupID
'D' = Delete
'E' = Export

"*" = All Entries (public or private)
"&" = only private entriesseules
""  = only public entries
```

2.2. **"U"** Operation on an individual user
- **"D"** Delete
- **"E"** Exports to a file with the name of the glossary and the session number

```
59, XX2XX  UDE wkg1  EN FR #%nom%#
```

2.3. **A"** Adding a Glossary
- "**GloFileName.txt "** file name of the glossary to be added

Details: Three possible functions, "**G**" = WorkGroupIds, "**U**" = Users, "**A**" = Add a glossary. Additional parameters depend on the selected function. **"DE"** for **"G"** or **"U"**, and a file name for **"A"**. The server returns a session number so that you can follow its execution in the background.

Java code:

```
String Line, Progress, Params;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
Line = Connector.WaitResultString();
If (Line.equal("IsAdmin=1")) {
    Params = "GlosName" + #9 + "GDE" + #9 + "*" + #9 + "EN" + #9 + "FR";
    Connector.PostCommand(Consts.cmdPurgeGlo, Params);
    Line = "Indicators=" + Connector.WaitResultString();
    Do {
        Connector.PostCommand(cmdGetParam, Line);
        Progress = Connector.WaitResultString();
    } While ((Progress.indexof("Msg=0") >= 0);
}
```

cmdGetParamSession = 61

Retrieves one or more parameters of a specific or global session. ~ Restricted to Administrators

Parameters: parameters are separated with a Tab character (# 9)

1: Session number

2: Parameters as defined in the command ***cmdGetParam***

Details: WfServer searches the session given by the number. If this session exists, the following parameters are retrieved from the session.

Java code:

```
String Line;
int ErrorNumber ;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
If (Line.equal("IsAdmin=1")) {
    Line = "5678" + #9 + "FuzzyThreshold"
    Connector.PostCommand(Consts.cmdGetParamSession, Line);
    Line = Connector.WaitResultString();
    ErrorNumber = Connector.Status.Msg;
}
```

cmdConfirmation = 63

Sends the confirmation code. ~ Restricted to Administrators

Parameter: Confirmation code on 10 alphanumeric characters.

Details: This command is not sent alone, it's a confirmation of an order previously sent to the server. At the end of the previous command, WfServer sends the customer a confirmation string (10 alphanumeric characters) to be returned by the client application.

Java code:

```
ses the example in command cmdShutDown.
```

cmdShutdown = 64

Request a VLTM server shutdown. ~ Restricted to Administrators

Parameters: None.

Details: WfServer sends the user a confirmation string (10 random alphanumeric characters) before executing the command. The distant user will have to return this confirmation string before the maximum time of the execution of a command.

Java code:

```
String Line, ConfirmationStr;
int ErrorNumber ;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
Line = Connector.WaitResultString();
If (Line.equal("IsAdmin=1")) {
    Connector.PostCommand(Consts.cmdShutDown, null);
    ConfirmationStr = Connector.WaitResultString();
    If (Connector.Status.Msg = msgConfirmation) {
        Connector.PostCommand(Consts.cmdConfirmation, ConfirmationStr);
    }
    ErrorNumber = Connector.Status.Msg;
}
```

cmdRestart = 65

Request a restart of the VLTM server. ~ Restricted to Administrators
Parameters: None.

Details: WfServer sends the user a confirmation string (10 random alphanumeric characters) before executing the command. The distant user will have to return this confirmation string before the maximum time of the execution of a command.

Java code:

```
String Line, ConfirmationStr;
int ErrorNumber ;
TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(Consts.cmdSetParam, "IsAdmin=" + AdminPassword);
Line = Connector.WaitResultString();
If (Line.equal("IsAdmin=1")) {
    Connector.PostCommand(Consts.cmdRestart, null);
    ConfirmationStr = Connector.WaitResultString();
    If (Connector.Status.Msg = msgConfirmation) {
        Connector.PostCommand(Consts.cmdConfirmation, ConfirmationStr);
    }
    ErrorNumber = Connector.Status.Msg;
}
```

cmdSearchContextExt = 66

Lucene-compatible filtered search ~ No restriction.
Parameters:

Three lines, each separated by \ r (# 13) or tab (# 9)

1- The **first line** consists of parameters associated with values, separated by ";".

- Search options:

"CaseSensitive",	"0", "1"	default = 0
"MatchWholeWords",	"0", "1"	default = 0
"PageOffset",	"0".."32767"	default = 0
"PageLength",	"0".."32767"	default = 50
"IndexUsed"	"[0,1,2]"	default = 0

- Filter conditions:

"OnlyPublicTus",	"0", "1"	défaut = 0
"CreationDate",	"20080403~163538"	
"CreationUser",	"Foobar"	
"Workgroup",	"1234567890" (10 caracters)	
"Attr1", "Attr2", "Attr3", "Attr4"	character string	
"Occur",	"0".."2147483647"	
"Source",	character string	
"Target",	character string	

- Operators :

"=", "<", ">", "<=", ">=", "<>", "Э", "Є",
"!=", "!<", "!>", "!Э", "!Є"

- Example :

"IndexUsed=0;CaseSensitive=1;PageOffset=0;CreationDate>=20080403~163538;CreationUser=Foobar;Attr1Єjohn;Occur>10"

2- The **second and third** lines respectively contain the source and target queries, even if they are empty.
The query consists of **keywords** (or **groups** of keywords) surrounded by wildcards (+) (-), (*), which refine the search.
A **group** of keywords is framed in quotation marks "".
The symbols (+), (-) in front of a keyword (or group) indicate that the keyword **must** (+) or **should not** (-) appear.
(*) when placed at the end of keyword this symbol indicates that all the terms with identical root accepted.

Examples:

+token   -token*   +"segment token" -"segment* token*"

Details: The results are ranked in order of relevance, most relevant first.

Java code:

```
String Line =
"IndexUsed=0;CaseSensitive=1;PageOffset=0;CreationDate>=20080403~163538;CreationUser=Tar
tempion;Attr1Єjohn;Occur>10\n-renai* +\"the hotel\"";

TMServerTCPConnector Connector = new TMServerTCPConnector();
Connector.PostCommand(cmdSearchContextExt, Line);
If Connector.WaitLastTicket().Msg = msgEndJob {
    Line = Connector.WaitResultString();
}
```

Appendix 4: WfServer API error codes

msgInProgress	0	Command in progress
msgEndJob	1	Command done ("No error")
msgBreaked	2	Command was interrupted
msgNotStarted	3	The timeout to execute this command is exceeded
msgTMNotAssigned	4	Command not started
msgTMNotIndexed	5	TM not assigned to account
msgTMNotOpen	6	TM must be reorganized
msgNotConnected	7	Database is not opened
msgInvalidLicense	8	License is not valid
msgInvalidWkSpace	9	WorkSpace not supported
msgInvalidClient	10	Unknown client
msgAlreadyInUse	11	Already in use
msgNotOnSelf	12	Cannot perform action on yourself
msgNotAdmin	13	You are not an Administrator

msgNetCmdInterdit	14	Internal command
msgNotAlone	15	You are not alone
msgInterdit	16	Not authorized
msgAlreadyConnected	17	Already connected
msgErrTCPIP	18	Error TCP/IP
msgTMLocked	19	TM is Locked
msgMemCrash	20	Crash memory
msgTMUsedProcess	21	TM used by process
msgTMNotExist	22	TM not exist
msgMaxUserDone	23	Max users done
msgMustBeCrypted	24	Only in encrypted mode
msgUserDisabled	25	User disabled
msgBadPassw	26	Bad password
msgAccountNotDefined	27	Account not defined
msgWkSNotDefined	28	WkSpace not defined
msgServerFull	29	FIFO Server full
msgNotExist	30	Not exist
msgAlreadyExist	31	Already exist
msgInvalidParam	32	Invalid parameter(s)
msgTUnotDeleted	33	TU not deleted
msgBadPointer	34	Bad pointer
msgOldIndex	35	Old Index
msgIsNotTU	36	Not a valid TU
msgServerLocked	37	Server is locked
msgNoLangID	38	Language pair not defined
msgAccountNotActived	39	Account not actived
msgGlossaryAlreadyOpen	40	Glossary file already open
msgGlossaryNotExist	41	Glossary file does not exists
msgTimeOut	42	Time out
msgBadURL	43	Bad URL
msgNoGroupID	44	Workgroup ID not defined
msgWkgIdOrLangViolation	45	Workgroup ID or Language violation
msgGlossaryNotIndexed	46	Glossary is not indexed or reorganized
msgGlossaryLocked	47	Glossary is locked
msgGlossaryNotOpen	48	Glosary is not open
msgGlossaryNotDefined	49	Glossary is not defined
msgGlossaryNotAssigned	50	Glossary is not assigned to account
msgNotInLiteMode	51	Forbidden in demo mode
msgInvalidEntry	52	Invalid entry
msgConfirmation	53	Confirmation request
msgInvalidCmd	54	Invalid command
msgInsuffisantParameters	55	Insufficient parameters
msgNoTargetIndex	56	Server version does not support target index

Appendix 5: The Wordfast Server TM format

WFS uses the Wordfast Classic (WFC) TM format. A WFC translation memory is a tab-delimited text file. It's the simplest of all formats - it can be opened with text editors, like Notepad, or unicode-compliant word processors, as well as with Excel. WFC TMs can be regular ANSI (8-bit) text, or Unicode UTF-16 (both little-endian and big-endian).

A Translation Memory (TM) is a set of lines (paragraphs) of text. In a pure text file where the display does not wrap, lines *are* paragraphs. The very first line is a header, and all other lines are Translation Units (TUs), sometimes called "entries". Lines/Entries/TUs are sets of fields, a field being any text (even lack of text, which denotes an empty field) followed by a tabulator. In other words, the WFC TM format is *Tab-delimited Text*, which is arguably one of the oldest, most robust, open, easy to manipulate data format ever. In the header (the very first line in a TM), each field begins with a % (per cent) mark.

Fields making up a TU:

Field	Example	Format	Remark
Date	20041231~165410	yyyymmdd~hhmmss - the example here means 31 December 2004, at 16:54:10, local time. See note on the tilde ~ character further below.	Optional field: can be empty
User ID (Attribute #1)	YAC	Initials of the TU's creator.	Optional field: can be empty
Counter	5	A number between 0 and 9999 that records how many times this TU was proposed as a 100% match and accepted, meaning, re-used, *as it is*.	Optional field: can be empty
Source language	EN-US	TMX-compliant language code (but case-insensitive with WFC). It is made of a two-letter ISO language code, and optinally, a dash followed by a two-letter local variant.	Optional field: can be empty. Rule: field cannot be longer than 5 characters.
Source segment	Red Riding Hood was walking in the woods.	The source segment. Maximum size: 8000 Unicode characters.	Should contain at least one printable character.
Target language	FR-FR	Language code, TMX-compliant	Optional field: can be empty. Rule: field cannot be longer than 5 characters.
Target segment	Le Petit Chaperon Rouge se promenait dans les bois.	The target segment. Maximum size: 8000 Unicode characters.	Optional field: can be empty
Attribute #2 (optional)	EL	A mnemonic (maximum length=64 characters; no space allowed) for user-defined attributes. See WFC's "Sample" attributes for typical values, for example, client, domain, job number, department, etc.	Optional field: can be empty+tabulator omitted
Attribute #3 (optional)	PS		Optional field: can be empty+tabulator omitted
Attribute #4 (optional)			Optional field: can be empty+tabulator omitted
Attribute #5 (optional)			Optional field: can be empty+tabulator omitted

Here are the first two paragraphs (the TM's header and first Translation Unit) of a TM where the TU is defined as in the table above. Paragraphs are long, so they may wrap in your display - but there are only two paragraphs:

```
%20041231~160445	%YAC, Yves A. Champollion	%TU=00000000	%EN-US	%WFC TM v5.0	%FR-FR
	%87412764
```

```
20041231~165410	YAC	5	EN-US	Red Riding Hood was walking in the woods.	FR-FR	Le Chaperon Rouge se promenait dans les bois.	EL	PS
```

When reading a TU, WFC defaults on the side of optimism in case the TU does not look correct or canonical. When in a TU:

- *the date is missing*: if WFC is executing a loop that parses TUs, then it will take the previous TU's date and increment it with one second, otherwise, it will take the local machine's current date and time;
- *the user ID is empty*, WFC will assume the TM header's user ID. If it is missing, WFC will use the user's identity as defined in Ms-Word. If it is missing, WFC will use XX;
- *a language code is missing or incorrect - but less than 6 characters*: WFC will use the current TM's header language code (the code in the first line of the TM).

Fault detection A faulty line or TU is determined by counting how many tabulators are in a line of text. A line of text with less than 6 tabulators cannot form a valid TU. Another fault-detection method used by Wordfast is that language codes should not be longer than 5 characters. When language codes of more than 5 characters are encountered during a TM reorganisation, it is an indicator that something is amiss with that particular TU, and it is assumed to be faulty. Most Wordfast programs do not halt on faulty TUs, they simply ignore them.

Remarks:

1. The date does not necessarily have a tilde (~) separating date and time. Any printable character can be used there, except a number. WFC uses the tilde (~), and the equal (=) sign. The equal sign means the TU was "marked" (flagged) by WFC's data editor. This has no consequence at all on the TU's status: it remains fully valid. Although WFC always records the date and time when writing a TU, the date and time are optional and could be empty (or even made of an invalid date) in which case WFC would simply assume the current computer's date and time. All dates and times are "local", taken from the local computer's clock.
2. If any optional field is left empty, its trailing tabulator should be present. For a TU to be valid, there must be at least six tabulators, with the fifth field (the source segment, located between the fourth and the fifth tabulator) made of at least one printable character.
3. The date's first character (a number from 0 to 9, usually, a number 2 if the TU was created in the current millenium) can appear to be "x". This means that this TU is not valid anymore. The first full reorganisation of the TM by WFC will erase this TU. Do not remove the "x", or replace it with a number, unless you know what you are doing.

Placeholders as tags

Placeholders are used to encapsulate tags (that contain formatting information), or a few special characters. A WFC placeholder always has the following format: `&tX;` where X can take various values: `&t=;` `&tA;` `&t1;` `&t#;` , etc.

`&t1;`	a placeholder for a Word graphic;
`&t2;`	a placeholder for a Word footnote/endnote;
`&t9;`	a tabulator mark;
`&t#;`	a manual line feed;
`&tA; &tB; &tC; ... &t¥;`	constitute 100 placeholders for tags;
`&t=<some tag="here">;`	records an "unknown" tag. Unknown tags are found only in a target segment, but not in the matching source segment. Colons in the tag are escaped with a backward slash \.

Note to engineers
The ampersand, quote, greater, smaller (& "< >) characters are not escaped. The WFC TM format is not a member of the SGML/HTML/XML family.

Limitation: A WFC TM would create a slightly fuzzy match with text containing `&tX;` as literal text, as in this very paragraph. That is a minor and non-lethal limitation, which, to our knowledge, has not happened in a decade.

Tags in a WFC TM

When dealing with so-called *tagged documents*, a WFC TM records placeholders for tags. Those placeholders have a `&tX;` format, where X is the order of appearance of tags in the source segment. The X order is noted A (ANSI decimal 65), B, C, etc., up to ANSI decimal code 165. Thus, there can be no more than 100 tags in a WFC segment.

For example, the following tagged source segment:

```
<FONT FACE="Helvetica">This is some text.</FONT>
```

would appear, in a WFC TM as:

```
&tA;This is some text.&tB;
```

At translation time, when WFC pulls a TU from the TM and is about to propose the TU's target segment as a translation candidate, WFC uses a substitution algorithm to dress the proposed target segment with the full "real" tags, taken from the *document*'s (not the TM's) source segment, using a triangulation method:

Document's source segment <—> TM's source segment <—> TM's target segment

The triangulation can be successful only if all target tags have a "parent" tag in the source segment. This is because, at translation time ("leverage" time), only the new source segment, and the target has to be worked out by the machine. In other words, it's not a problem if the TM's source segment contains tags that do not appear in the TM's target segment. The reverse is a problem, however. If the TM's target segment has tags that do not appear in the TM's source segment (*orphaned* tags), WFC records the full syntax of these orphaned tags at TU *creation time,* so that they can be restored properly at *translation time*, when the target segment must be proposed with the correct format. If we have, at TU creation time:

In source segment:	`<FONT FACE="Arial">This is some text:`
In target segment:	`<FONT FACE="Arial">Voici du texte&nbsp;:`

then the target segment would be recorded in the TM as:

```
&tA;Voici du texte&t=&nbsp\;;:
```

where `&t=` opens the original tag syntax (`&nbsp;` in our example) and `;` (colon) closes the sequence.

Other examples of segments:

In source segment:	`<FT>This is some text<AR> here<FT>.`
In target segment:	`<AR>Voici du texte<FT> ici.`
In TM TU source:	`&tA;This is some text&tB; here&tA;.`
In TM TU target:	`&tB;Voici du texte&tA; ici.`

In source segment:	`<FT>This is some text<AR> here.`
In target segment:	`<AR>Voici du<AR> texte<X;X> ici<FT>.`
In TM TU source:	`&tA;This is some text&tB; here.`
In TM TU target:	`&tB;Voici du&tB; texte&t=<X\;X>; ici&tA;.`

In most translation memory systems, TMs are bloated with tags that *do not belong there*. Engineers may have overlooked that a TM takes significance and value when its content is put to use, meaning, when its past translations are *leveraged* for a new transation project. The point here is, leveraging TM content is done in the presence of a new document to be translated. At that point, the program can operate a triangulation between a new document's source segment which contains *the new*

formatting, and an existing TM source/target pair which contains previous formatting placeholders.
To make things worse for formats that are obsessed with recording full formatting information, the XML layer on top of the original format's layer (such as TMX recording RTF or richly formatted text) creates verbosity that borders on silliness.
Wordfast opted for an agile format with a footprint 10 to 15 times smaller than TMX.