WotWizard Manual

WotWizard builds, from the blockchain and sandbox of Duniter, a prediction of the future entries of candidates into the Duniter Web of Trust (WOT). It uses a simulation of the Duniter mechanism. When several possibilities may happen, each one is listed with its probability. The published lists are automatically updated every five minutes, and any change is signaled visually.
WotWizard contains several tools and a fast **Web of Trust Explorer**.

Warning: You must use the 1.7.17 version of duniter or any later version in the 1.7.x series.

How to use it?

This program needs a **Duniter** node running on the same computer.

This node must be configured for the use of WotWizard. To do so, edit its configuration file ("conf.json"; see *Configuration* below) in a text editor and put the value of the "storage.wotwizard" field to "true".

WotWizard includes a server executable (**wwServer**) written in *Go* (v1.13.6) and running under GNU / Linux. This server communicates by the way of disk files containing *GraphQL* requests on input, and JSON answers on output. WotWizard includes an optional graphical user interface too (**WotWizard.exe**) written in *Component Pascal* (BlackBox v1.7.1, running under Wine). This GUI creates GraphQL requests through menus items and interactive windows, and displays answers through texts or graphics.

wwServer is to be placed in an empty directory where it can be run. It creates itself the sub-directories and files it needs, all of them being included in the inital directory. If the Duniter database is not at its standard place, you can give it with a command line option.

WotWizard.exe runs natively under Windows. In order to make it work with wwServer, place it in the same directory. Once *wine* installed, you can run it.

Server Mode

You can use WotWizard as a server only, if you want to display its results yourself, e.g. in a web page. Configure first Duniter for WotWizard (see above). If Duniter installation is standard, (its database must lie in the file "~/.config/duniter/duniter_default/wotwizard-export.db"), you can run wwServer without options:

```
$./wwServer
```

Otherwise, on the first run, add the -du option:

```
$./wwServer -du <path to Duniter database>
```

On later runs, the -du option is no more useful.

It's also possible to modify this path by editing the "rsrc/duniter/init.txt" file.

The amount of memory allocated by wwServer for the calculus of its predictions is limited. Its default value is 430,000,000 bytes. You can change it by editing the "rsrc/duniter/parameters.txt" file. The larger it is, the more precise are WotWizard predictions, but if it's too big, wwServer may crash.

Any change in "init.txt" or "parameters.txt" is taken into account only when wwServer (re)starts.

After a not-so-long initialization phase (which starts after the recording of a new block in Duniter) that you can follow in the "rsrc/duniter/log.txt" file, request files can be deposited into the "rsrc/duniter/Work/Query" sub-directory. Their names are meaningless and you can deposit as many of them as you want. They are immediately read and destroyed. After their handling, answer files appear in "rsrc/duniter/Work/Json" sub-directory; see below the rules between requests ans answers. In the case of errors in request files, error messages appear in log.txt and Stderr.

To stop wwServer, press Ctrl-C or use any other standard method.

The WotWizard database is contained in the "rsrc/duniter/System/DBase.data" file. At every start of wwServer, a new copy of DBase.data is created. At most, two copies are saved: "Dbase.data.bak" is the last one and "DBase.data.bak1" is the previous one. If you have a doubt on the integrity of the datebase, replace it by renaming an older copy.

List of GraphQL requests, and their meanings

{WWServerStart}: Install a subscription to the update of the WotWizard window. Do nothing if the subscription is already installed (but may change the name of the output JSON file, see below).

{WWServerStop}: Erase the subscription to the update of the WotWizard window. Do nothing if the subscription is not already installed.

{WotWizardListFile}: Display the WotWizard file.

{WotWizardListPerm}: Display the list of WotWizard permutations.

{IdSearchFind(Hint:"<hint>"){<set of (OldMembers, Members, FutureMembers)>}}: Display the list of identities whose pseudos or public keys begin with <hint> and whose status is old member, active member or newcomer, according to the given set; the list includes pseudos, hashes and status.

{IdSearchFix(Hash:"<hash>"){<set of (Distance, Quality, Centrality)>}}: Display informations for the identity corresponding to <hash> and including, or not, her distance to the wot, her quality and her degree of centrality according to the given set.

{History(Uid:"<uid>")}: Display the history of wot entries / exits of a member or old member.

{Parameters}: Display the block 0 parameters of the money.

{IdentitiesRevoked}: Display the list of revoked identities.

{IdentitiesMissing}: Display the list of excluded, but not yet revoked, identities.

{IdentitiesMembers}: Display the list of active identities.

{CertificationsFrom}: Display the list of active certifications, grouped by senders.

{**CertificationsTo}**: Display the list of active certifications, grouped by receivers.

{Sentries}: Display the list of sentries.

{Sandbox}: Display the content of the sandbox, with different sortings.

{QualitiesDist}: Display the distances to the wot of all active members.

{QualitiesQual}: Display the qualities of all active members.

{CentralitiesAll}: Display the degrees of centrality of all active members.

{MemEnds}: Display the ends of validity of memberships for all active members, sorted by dates.

{MissEnds}: Display the dates of revocation of all excluded, but not yet revoked, identities, sorted by dates.

{CertEnds}: Display the dates of loss of the fifth received certification for all active or excluded, but not revoked, identities.

{MembersCountAll}: Display the number of active members, sorted by dates of events (in or out the wot).

{MembersCountFluxAll(timeUnit:<time unit (s)>)}: Display the flux of active members by <time unit>.

{MembersCountFluxPMAll(timeUnit:<time unit (s)>)}: Display the flux of active members by <time unit> and by member.

{MembersFirstEntryAll}: Display the number of first entries into the wot, sorted by dates of events (entries).

{MembersFEFluxAll(timeUnit:<time unit (s)>)}: Display the flux of first entries by <time unit>.

{MembersFEFluxPMAll(timeUnit:<time unit (s)>)}: Display the flux of first entries by <time unit> and by member.

{MembersLossAll}: Display the number of members exiting the wot, minus the number of reentries (losses), sorted by dates of events (in or out the wot).

{MembersLossFluxAll(timeUnit:<time unit (s)>)}: Display the flux of losses by <time unit>.

{MembersLossFluxPMAll(timeUnit:<time unit (s)>)}: Display the flux of losses by <time unit> and by member.

If an alias is added to a command (e.g. "wwResult" in {wwResult:WWServerStart}), it gives the name of the output json file (here wwResult.json); otherwise, this name is the name of the command itself (e.g. output of {WWServerStart} is in the file WWServerStart.json).

Interactive Mode

Place the "WotWizard.exe" file in the same directory as wwServer. Run WotWizard by a double-click or with the command line (from the same directory):

```
   $wine WotWizard.exe
```

WotWizard.exe doesn't start wwServer, that you must run before any request. It doesn't stop it either.

Once WotWizard has started, choose your language and, possibly, your prefered fonts, in "Edit -> Preferences...".

After that, you can run commands with menus items and see answers in opening windows.

The WotWizard Window

Open the WotWizard Window with "WotWizard -> New WotWizard View".

You can:
- choose the way the list is displayed (by names or by dates, or metadata)
- manually update the list (it's automatically updated every time a new block appears)

When the list has changed, two asterisks appear, one on each side of the title, and a new button "Check" is created. Click on the button to make the marks disappear. You can then compare the new and old lists (by Dates and Metadata) by clicking on the button "Compare", or by using the menu item "Edit -> Compare Texts" (keyboard shortcut: F9).

File: WotWizard calculates its forecasts from a state of affairs called *File*. It's a list of *dossiers* (newcomer alias + certifications list) and *internal certifications* (between members), sorted by availability dates. The command "WotWizard -> File" displays the following details:

- *Internal certification* : one line with:
  + the receiver and sender of the certification (aliases) with an arrow (←) between them;
  + the certification availability date;
  + between brackets, the validity limit of the certification;
  + a summary of the certification state: OK if it's available and valid, or KO otherwise.
- *Dossier* :
  + a first line describing the newcomer:
    * the number of main certifications (those which fix the availability date of the dossier, see below);
    * her alias;
    * between brackets:
      - her availability date;
      - its minimal availability date, two months (sigReplay) after her previous membership application;
    * between brackets, the validity limit of her application;
    * the distance rule result, as the percentage of reachable sentries;
    * a summary of the dossier state: OK if it's valid and if the rule of minimal certification number and the distance rule are verified, or KO otherwise (certification availabilities are not checked here).
  + one additional line for each external certification, in the same form as for internal certifications, but without the sender alias (which is the newcomer's); these certifications are sorted by availability date.

Main certifications: In a Dossier, it's the *n* first external certifications (in the order of availability dates), where *n* is the number of certifications it owns when it is less than or equal to *sigQty* (5). If it is greater, it's the number of certifications needed to satisfy the distance rule, or the total number if the distance rule can't be satisfied.

Permutations : The command "WotWizard -> Permutations" displays all entry ranks permutations predicted by WotWizard, with their probabilities. **Warning**: their number may be sometimes very big and the display very large!

Web of Trust Explorer

Open the WoT Explorer with the menu command "Web of Trust -> Explorer".

See the explorer map.

You can search any identity in the blockchain or the sandbox by typing its first characters or the first characters of its public key into the first top field and by clicking on the "Search" button. The possible identities appear in the list at the bottom: choose the one you want to look at. Some details appear in the "Identity" frame, and the certifications, received and sent, in the "Certifications" frame. You can look at the identity of one of the senders or receivers by clicking on the corresponding "Go" button; step back with the arrow buttons. You can also display a text of informations on certifiers or certified identities by clicking on one of the "D" buttons (list of present certifiers or certified identities, sorted by alphabetic order, by expiration dates of membership (or limit date before revocation if the identity is no more member - "×" before her name) and by expiration dates of certifications, and list of all non-revoked certifiers or certified identities of all times, able to certify or be certified again). Names of certifiers and certified waiting in sandbox are preceded by the symbol º.
Displayed dates are member's registration date into the blockchain, expiration dates, of member's membership and of her certifications, and the availability date of the next sent certification in the field "Availability" (if already available, this date comes after an exclamation mark "!"). If member's end of membership has been requested, her expiration date of membership is preceded by a cross (×). In this case, she can no more get new certifications, and, without renewal of her membership, she will be excluded at this date.

The degree of centrality *c* of a member is the number of oriented paths (certifier -> certified) of shortest lengths to which she belongs and whose length is, at most, *stepMax* (5) plus one steps from the certifier. The centrality level *c'* of a member is calculated from her degree of centrality *c* by the expression:

$$c' = 100 \frac{\ln(1+c)}{\max[\ln(1+c)]}$$

where $\max[\ln(1+c)]$ is the greatest value of $\ln(1+c)$ for all members, and where ln is the natural logarithm.

One may consider the *centrality level* of a member as the level of her involvement to help people she certified to respect the distance rule, independently of her capacity to do so. By opposition, the *quality* of a member is her capacity to help people she certifies to respect the distance rule, independently of her level of involvement.

A member with a quality greater than or equal to *xpercent* (= 80%), makes, only by herself, people she certifiies respect the distance rule.

Tools

Parameters (*Parameters*): Display the basic parameters of money.

Revoked Identities (*IdentitiesRevoked*): Display revoked identities with their aliases and public keys, and the dates of their entries.

Missing Memberships (*IdentitiesMissing*): Display identities whose memberships were'nt renewed, with their aliases and public keys, the dates of their entries and their limit dates before revocation.

Identities (*IdentitiesMembers*): Display identities of all members with their aliases and public keys, the dates of their entries and their limits of validity.

Certifications From... (*CertificationsFrom*): Display all certifications in the blockchain, sorted by senders, with their inscription dates.

Certifications To... (*CertificationsTo*): Display all certifications in the blockchain, sorted by receivers, with their inscription dates.

Sentries (*Sentries*): Display the identities of sentries.

Sandbox (*Sandbox*): Display identities and certifications in sandbox;

1) Identities sorted by hashes, with hash, public key, id & expiration date
2) Identities sorted by public keys, with public key & hash
3) Identities sorted by ids, with id & hash
4) Certifications sorted by senders' public keys, with sender's public key, receiver's hash & expiration date
5) Certifications sorted by receivers' hashes, with receiver's hash, sender's public key & expiration date

Distances (*QualitiesDist*): Give the distances of all members to the web of trust (proportion of sentries reachable in at most *stepMax* (5) steps) sorted by proportions (with drawing) and by aliases.

Qualities (*QualitiesQual*): Give the qualities of all members sorted by qualities (with drawing) and by aliases.

Centralities (*CentralitiesAll*): Give the centrality levels of all members sorted by centrality levels (with drawing) and by aliases.

Limits of Memberships (*MemEnds*): Display the validity limits of memberships for all members, sorted by dates.

Limits of Missing Memberships (*MissEnds*): Display the limit dates before revocation of identities with not-renewed memberships, sorted by dates.

Limits of Certifications (*CertEnds*): Display the validity limits, for lack of certifications, of memberships for all members, sorted by dates.

History

Number of Members (*MembersCountAll*): Give the list of block dates where the number of members changed and the corresponding members' number since the beginning of the money (with drawing).

Flux of Members (*MembersCountFluxAll*): Changes in the number of members by time unit (i.e.: one month). So, it's the derivative of the number *n* of members with respect to time t : $\frac{\mathrm{d}n}{\mathrm{d}t}$ (with drawing).

Flux of Members per Member (*MembersCountFluxPMAll*): Flux of members divided by the number *n* of members: $\frac{\mathrm{d}n}{n\,\mathrm{d}t}=\frac{\mathrm{d}\ln(n)}{\mathrm{d}t}$ (unit: percent by time unit). With drawing.

First Entries (*MembersFirstEntryAll*): Give the list of block dates where the number of members entering the WoT for the first time changed and the corresponding number *e* of first entries since the beginning of the money (with drawing).

Flux of First Entries (*MembersFEFluxAll*): Changes in the number of first entries by time unit (i.e.: one month). So, it's the derivative of the number *e* of first entries with respect to time t : $\frac{\mathrm{d}e}{\mathrm{d}t}$ (with drawing).

Flux of First Entries per Member (*MembersFEFluxPMAll*): Flux of first entries divided by the number *n* of members: $\frac{\mathrm{d}e}{n\,\mathrm{d}t}$ (unit: percent by time unit). With drawing.

Losses (*MembersLossAll*): Give the list of block dates where the number of members changed and the corresponding differences $l=e-n$ between the number *e* of first entries and the number *n* of members since the beginning of the money (with drawing).

Flux of Losses (*MembersLossFluxAll*): Changes in the losses by time unit (i.e.: one month). So, it's the derivative of the losses *l* with respect to time t : $\frac{dl}{dt}$ (with drawing).

Flux of Losses per Member (*MembersLossFluxPMAll*): Flux of losses divided by the number *n* of members: $\frac{dl}{n\,dt}$ (unit: percent by time unit). With drawing.

Use it and enjoy! - ¡Úsalos y disfrútalos! - Bonne utilisation - Приятного использованиян - Powodzenia - Viel Spaß

Gérard Meunier