SynchroSocket User Guide

Version 0.12

28 September 2026

Contents

1. What SynchroSocket does

SynchroSocket reports a host's **network connections** to SynchroServer. Every few seconds it reads the socket table, groups the connections into categories, and sends one set of counts per category.

It answers the questions a host's CPU and memory metrics cannot:

- How many clients are connected to this service right now?
- Is anything queueing on the receive or send side?
- Who is connecting to us, and who are we connecting to?
- Is a port still listening?

It is a companion to SynchroAgent, not a replacement. SynchroAgent measures the host; SynchroSocket measures its conversations.

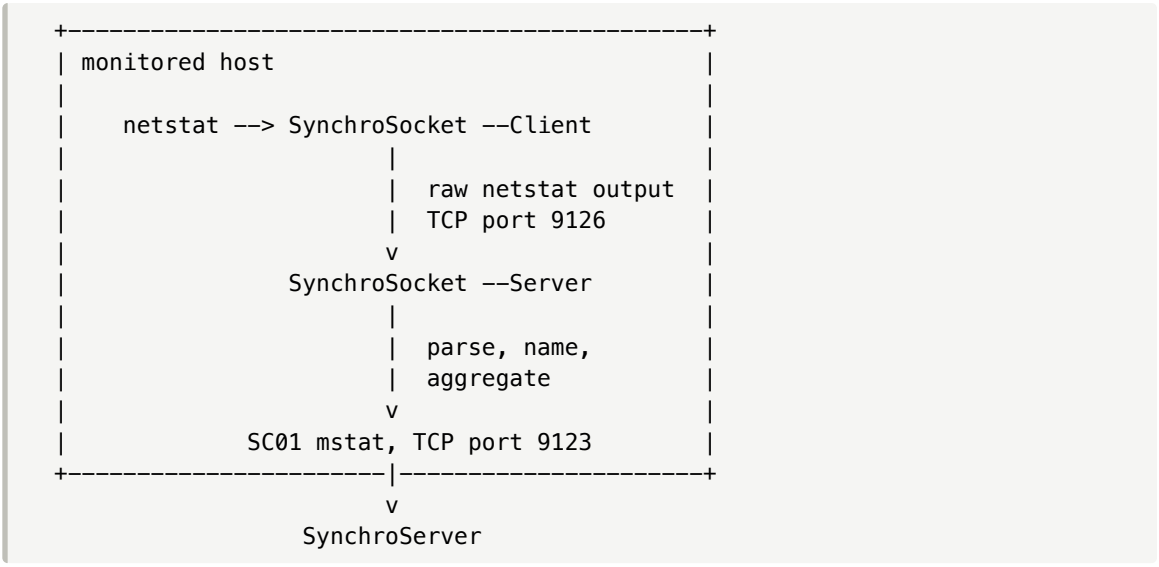
1.1 The cardinality problem

A connection monitor that reports raw addresses destroys itself. Every peer address it has ever seen becomes a permanent time series, and an internet-facing host meets thousands of port scanners a day. Measured on one production host: **2,840 targets, of which 2,575 were scanners hitting port 22, and only 33 had been written to in the previous hour.** The other 2,807 were dead series consuming disk and cluttering every target list, forever.

Naming is therefore not cosmetic. It is the mechanism that keeps the metric namespace bounded. Sections 4 and 5 are the heart of this guide for that reason.

2. How the pieces fit together

SynchroSocket runs as a **pair of processes**, usually on the same host.



Process	Responsibility
`--Client`	runs `netstat` every *n* seconds, ships the raw output to the server
`--Server`	parses, names and aggregates it, forwards metrics to SynchroServer

The split exists so that **one server can serve many clients**. The client is deliberately thin: it does no parsing and holds no configuration beyond where to send its output. All naming and aggregation happens once, on the server, which means a naming change takes effect everywhere by restarting one process.

The client builds its command from the collection settings, for example `netstat -atunW --numeric-port`.

2.1 What goes on the wire

The client sends raw `netstat` text. The server sends Synchro **SC01 mstat** lines, tab-separated:

```
SC01<TAB>mstat<TAB>20260928.142919<TAB>affa5_SYNCHROSOCKET_COUNT_tcp-LISTEN-ssh=4
```

3. What it measures

Keys follow Synchro's four identifiers, `COMPONENT_TYPE_METRIC_TARGET`.

Metric	Meaning
`COUNT`	connections in this category
`RECVQByte`	total receive-queue bytes in this category
`SENDQByte`	total send-queue bytes in this category
`HeartBeat`	server liveness, emitted every 5 seconds
`MODE`	`Client` or `Server`, so you can see what is running where
`VERSION`	the running version, as `V-0.12`
`SIZEByte`	size of the netstat payload (client, with `--ClientMetrics`)

`RECVQByte` and `SENDQByte` are the early-warning pair. A receive queue that stays above zero means the application is not reading as fast as the network is delivering; a send queue that does the same means the far end is not reading, or the path is congested.

3.1 The three shapes of a target

```
tcp-LISTEN-ssh                          a listening port
EXT-tcp-ESTABLISHED-live5-synchro-stat  an inbound connection matching a rule
OUT-tcp-TIME-WAIT-internal-ssh          an outbound connection matching a rule
```

Reading the fields left to right:

Field	Example	From
direction	`EXT`, `OUT`, or absent	which rule list matched
protocol	`tcp`, `tcp6`, `udp`	netstat
state	`ESTABLISHED`, `LISTEN`, `TIME-WAIT`	netstat
peer	`live5`, `internal`, `other`	`HostNames` (section 4)
port	`ssh`, `synchro-stat`, `8443`	`PortNames` (section 4)

Targets without a direction prefix are the **global view**: every connection on the host, grouped by protocol, state and local port, regardless of peer. This is the view that tells you a port is listening and how busy it is. The `EXT-` and `OUT-` targets come from the connection rules in section 6 and add the peer dimension.

4. Naming hosts and ports

Naming turns `10.0.3.14` into `internal` and `9123` into `synchro-stat`. Two tables do the work.

```
"PortNames": {
      "22": "ssh",
      "9123": "synchro-stat"
},
"HostNames": {
      "51.79.18.22":    "live5",
      "10.0.0.0/8":     "internal",
      "192.168.0.0/16": "internal"
}
```

`HostNames` accepts exact addresses, hostnames and **CIDR networks**, for IPv4 and IPv6 alike. Networks are matched **most-specific-first**, so a `/32` beats the `/16` it sits inside, which beats the `/8` around that. This is what lets you write one broad rule and then carve exceptions out of it:

```
"10.0.0.0/8":    "internal",
"10.1.0.0/16":   "datacentre-a",
"10.1.0.7":      "database-primary"
```

An address of `10.1.0.7` resolves to `database-primary`, `10.1.9.9` to `datacentre-a`, and `10.20.0.1` to `internal`.

Several entries may share a name. Mapping three private ranges all to `internal` is the normal case, and collapses them into a single series.

4.1 What happens to everything else

`UnknownHost` and `UnknownPort` decide the fate of anything not in the tables.

Value	Effect
`name`	use the raw address or port number
`other`	collapse everything unnamed into a single bucket
`drop`	do not report it at all

`UnknownHost` defaults to `other`, and should usually stay there. This is the setting that solves the cardinality problem in section 1.1: every scanner on the internet lands in one `other` series instead of minting three of its own. You lose the ability to distinguish one unknown peer from another — which was never information you wanted — and you keep the count, so a spike in scanning is still visible.

`UnknownPort` defaults to `name`, the opposite choice, because port numbers are bounded and meaningful. An unexpected listener on port 4444 is something you want to see, not something to hide in a bucket.

Use `drop` when a category is pure noise and you do not even want the count.

5. Keeping the tables in their own files

Naming tables outgrow a configuration file. One estate-wide table is easier to maintain, review and copy between hosts than the same hundred entries pasted into every host's config — and pasted copies drift.

Either table can live in its own JSON file, named in the configuration:

```
"PortNamesFile": "portnames.json",
"HostNamesFile": "hostnames.json"
```

or on the command line:

```
bin/SynchroSocket-Linux --configFile conf/config-affa5.json --Server \
      --PortNamesFile /etc/synchro/portnames.json \
      --HostNamesFile /etc/synchro/hostnames.json
```

5.1 Two accepted file shapes

A name file may hold the mapping **on its own**:

```
{ "22": "ssh", "443": "https" }
```

or **wrapped** under the same key the configuration uses:

```
{ "PortNames": { "22": "ssh", "443": "https" } }
```

Both load identically. The wrapped form exists so a table can be lifted out of a configuration file and dropped into a new one unchanged, with no reformatting step in which to introduce a mistake.

5.2 Resolution order

A table can be assembled from up to three sources. They are applied in order of **increasing specificity**, and the later one wins on a conflict:

1. the inline `PortNames` / `HostNames` table in the configuration file
2. the file named by `PortNamesFile` / `HostNamesFile` in the configuration
3. the file given by `--PortNamesFile` / `--HostNamesFile` on the command line

Merging, rather than replacing, is the useful part. A shared file can carry the estate-wide table while an individual host keeps a handful of local entries inline, and a command-line file can override both for a one-off investigation without editing anything.

Relative paths resolve against the working directory, as `--configFile` does.

5.3 A missing file is fatal

A name file that is named but **missing, malformed, or not an object** stops the process at startup with a specific message and a non-zero exit status.

This is deliberate. The alternative — logging a warning and carrying on — would rename every peer to `other` and every port to a number. The result looks exactly like a quiet monitoring gap, and would be investigated as one, when the actual cause is a typo in a path. A process that refuses to start is noticed in seconds; a process that starts and lies is noticed in weeks.

```
Cannot read PortNames file "portnames.json" – [Errno 2] No such file...
File "portnames.json" should contain valid JSON format – Expecting value...
File "portnames.json" should contain a PortNames object, or the mapping on its
own
```

6. Connection rules

The global targets in section 3.1 ignore the peer. Connection rules add it back, for the traffic worth watching that closely.

Two lists, differing only in which port they match on:

List	Prefix	Matches on
`ExternalConnections`	`EXT-`	the peer's address and the peer's port
`OutgoingConnections`	`OUT-`	the peer's address and the **local** port

```
{
      "RuleName": "PortSSH",
      "Active": true,
      "ExternalIP": ".*",
      "ExternalPort": "^22$",
      "Label": "ssh-inbound",
      "GroupBy": "peer",
      "UnknownHost": "other"
}
```

Key	Default	Meaning
`RuleName`	—	identifies the rule
`Active`	`false`	rules are opt-in
`ExternalIP`	—	regular expression against the peer address
`ExternalPort` / `InternalPort`	—	regular expression against the port
`Label`	`RuleName`	name used when `GroupBy` is `rule`
`GroupBy`	`peer`	`peer` for one series per named peer, `rule` for one per rule
`UnknownHost`, `UnknownPort`	inherited	override the global policy, for this rule only

`Active` defaults to `false` so that a rule added to a shared configuration does nothing until someone turns it on deliberately.

`GroupBy` and the per-rule policy overrides are how one deployment holds both positions at once: keep per-host detail for a database tier, where the set of peers is small and knowing which one matters, and collapse everything facing the internet, where it does not.

7. Key safety

A Synchro key is split on _ and must yield exactly four fields. Every field SynchroSocket emits is therefore cleaned: characters outside `A–Z a–z 0–9 @ .` – become –.

This matters more than it sounds. **TCP state names contain underscores** — `TIME_WAIT`, `CLOSE_WAIT`, `FIN_WAIT1`, `LAST_ACK`, `SYN_RECV`. Before cleaning, every connection in one of those states produced a five-field key that SynchroServer rejected outright.

The consequence on one production host: `TIME_WAIT` was the single most common socket state on the machine, and had never once been recorded. Nothing alerted, because a rejected key is not a failed key — it simply never arrives, and an absent series looks the same as a quiet one. The states now appear as `TIME–WAIT`, `CLOSE–WAIT` and so on.

The same cleaning applies to names you choose. A `HostNames` entry of `db_primary` is emitted as `db–primary`. Prefer hyphens when naming.

8. Configuration reference

8.1 Identity

Key	Default	Meaning
`SynchroServer`, `SynchroServerPort`	— / `9123`	where metrics are sent
`SynchroSocketServer`, `SynchroSocketPort`	— / `9126`	where clients send netstat output
`SynchroType`	`SYNCHROSOCKET`	the Synchro TYPE in every key

`SynchroType` is overridden by `--SynchroType`, so one configuration file can serve several deployments — including running a new version alongside an old one under a different TYPE while you compare them.

8.2 Collection

Key	Default	Meaning
`IncludeTCP`, `IncludeUDP`	`false`	protocols to parse
`IncludeIPV4`, `IncludeIPV6`	`false`	address families to parse
`FrequencySecond`	`5`	seconds between client samples
`ConvertIPNames`, `ConvertPortNames`	`false`	let `netstat` resolve names itself
`MaximumNameLenght`	`20`	truncate addresses to this length *(spelling kept for compatibility)*
`ExportToFile`, `ExportFileName`	`false`	also write every metric to disk as JSON

Leave `ConvertIPNames` off. Enabling it makes `netstat` perform a reverse DNS lookup per connection, which is slow, and produces names from DNS rather than from your tables — so it bypasses the cardinality control entirely.

`ExportToFile` writes a daily-rotated JSON file, useful for auditing exactly what was sent. Note the known issue in section 11: nothing prunes these files.

8.3 Naming

Key	Default	Meaning
`PortNames`	`{}`	inline port table
`HostNames`	`{}`	inline host table, exact addresses or CIDR
`PortNamesFile`	—	JSON file holding a port table
`HostNamesFile`	—	JSON file holding a host table
`UnknownHost`	`other`	`name`, `other` or `drop`
`UnknownPort`	`name`	`name`, `other` or `drop`

9. Command line

Flag	Meaning
`--configFile FILE`	JSON configuration file *(required)*
`--Server`	run in server mode
`--Client`	run in client mode
`--ClientMetrics`	also send client-side metrics
`--SynchroType TYPE`	override `SynchroType`
`--PortNamesFile FILE`	port table; merged over the configuration's
`--HostNamesFile FILE`	host table; merged over the configuration's
`--LogDir DIR`	log directory *(default `.`)*
`--DataDir DIR`	export directory *(default `./`)*
`--HostName NAME`	override the local hostname used as COMPONENT
`--MaxIddleTime SEC`	seconds before an idle client thread exits *(default 30)*
`--Debug`, `--Verbose`	more logging; `--Verbose` also prints to stdout
`--version`	print the version and exit

10. Deploying

10.1 What to install

The release ships a self-contained executable per platform. Nothing else is required – no Python, no packages.

File	Runs on
`bin/SynchroSocket-Linux`	Linux with glibc 2.28 or newer (RHEL 8+, Ubuntu 18.10+)
`bin/SynchroSocket-Windows.exe`	Windows Server 2016 / Windows 10 and newer, 64-bit

The Linux binary is built on RHEL 8 deliberately: a binary links against the glibc it was built with and runs on that version and newer, so building on the oldest supported distribution gives the widest reach.

10.2 A worked example

Unpack the release and the tree is ready to configure:

```
bin/SynchroSocket-Linux     the agent
conf/config-sample.json     copy this to make your own
conf/portnames.json         the port table
conf/hostnames.json         the host table
start.sh                    start both processes, or stop them
doc/  logs/  data/
```

This is the live deployment on `affa5`. Its `conf/config-affa5.json` names the two tables and sets the TYPE:

```
{
      "SynchroServer": "live5.synchro-tech.com",
      "SynchroServerPort": "9123",
      "SynchroSocketServer": "live5.synchro-tech.com",
      "SynchroSocketPort": "9126",
      "SynchroType": "SYNCHROSOCKET09",
      "PortNamesFile": "conf/portnames.json",
      "HostNamesFile": "conf/hostnames.json",
      "UnknownHost": "other",
      "UnknownPort": "name",
      "IncludeTCP": true, "IncludeUDP": true,
      "IncludeIPV4": true, "IncludeIPV6": true,
      "FrequencySecond": 5
}
```

`start.sh` runs the pair in the right order:

```
CFG=conf/config-affa5.json ./start.sh
CFG=conf/config-affa5.json ./start.sh stop
```

Or start them by hand, server first:

```
bin/SynchroSocket-Linux --configFile conf/config-affa5.json --Server \
      --LogDir logs --DataDir data &
bin/SynchroSocket-Linux --configFile conf/config-affa5.json --Client --
ClientMetrics \
      --LogDir logs --DataDir data &
```

10.3 Windows

The client works the same way on Windows, with two differences worth knowing before you read the numbers.

Recv-Q and Send-Q are always 0. Windows `netstat` does not report queue depths at all – the columns do not exist. `RECVQByte` and `SENDQByte` are still emitted, so the key set matches Linux, but on a Windows host they carry no information. The early-warning use described in section 3 is a Linux capability only.

State names are translated to their Linux spellings. Windows reports `LISTENING`, `SYN_RECEIVED`, `FIN_WAIT_1`; these are rewritten to `LISTEN`, `SYN_RECV` and `FIN_WAIT1` before they leave the client. This is deliberate: in a mixed estate the same condition should produce the same target name whatever the host runs, otherwise every chart and alarm needs two versions.

The translation happens on the **client**, so a Windows client reports correctly to a Linux server with no configuration on either side. Run the pair with `start.ps1`:

```
.\start.ps1          # start server and client
.\start.ps1 stop     # stop both
```

`start.ps1` is for trying it out: the processes belong to your session and end with it. For a real deployment register the pair as scheduled tasks running as `SYSTEM` with an at-startup trigger, so they survive a reboot and run with nobody logged in:

```
$dir = "C:\SynchroSocket"
$args = "--configFile conf\config-windows.json --LogDir logs --DataDir data"
$principal = New-ScheduledTaskPrincipal -UserId SYSTEM -LogonType ServiceAccount
```

```
-RunLevel Highest
$trigger   = New-ScheduledTaskTrigger -AtStartup

foreach ($m in "Server", "Client") {
      $action = New-ScheduledTaskAction -Execute "$dir\bin\SynchroSocket-
Windows.exe" `
            -Argument "$args --$m" -WorkingDirectory $dir
      Register-ScheduledTask -TaskName "SynchroSocket-$m" -Action $action `
            -Trigger $trigger -Principal $principal
}
```

11. Operating it

11.1 Confirming it works

The startup log states what was loaded. Check this line first:

```
Naming: 10 ports, 7 host entries (3 ranges); unknown host=other, port=name
```

If the counts are lower than the tables you wrote, a file is not being read.

Then confirm the client connected and is sending:

```
Connected to SynchroSocket
Ready for Processing
affa5 – 20260928 – 142644 – 31 – [0]
```

The trailing numbers are the metric count for that cycle and the queue depth. A **queue depth that climbs** means SynchroServer is not accepting — see 11.3.

11.2 What healthy output looks like

A reasonable host produces **tens of targets, not thousands**. The affa5 deployment settled at 31. If you see hundreds, `UnknownHost` is set to `name` somewhere, or a rule is grouping by peer on internet-facing traffic.

11.3 Troubleshooting

Symptom	Likely cause
Process exits at startup	A named table file is missing or malformed — read the last log line
Target count in the thousands	`UnknownHost: name` on internet-facing traffic
Peers all named `other`	The host table is not loading, or does not cover your ranges
Ports appear as numbers	Expected for ports not in the table with `UnknownPort: name`
A state never appears	Pre-0.9 key-splitting bug; see section 7
Metric count steady, nothing in Synchro	Check the SynchroServer connection, not the naming

Renaming stops the old series. When you name a port that previously reported as a number, `tcp-LISTEN-111` stops and `tcp-LISTEN-rpcbind` begins. The old series is not migrated

— it simply goes quiet, and its history stays under the old name. Expect a discontinuity in any chart spanning the change.

Note also that SynchroServer flushes to disk on an interval of several minutes. A new target can be visibly emitted by SynchroSocket some minutes before its file appears on the server. Check the export in `DataDir`, not the server's files, when verifying a naming change.

11.4 Known issues

LISTEN state is shared across hosts. The listening-port table is keyed on protocol and port with no component, so a port listening on one monitored host can cause another host's outbound connections from the same local port to be counted as listening. Affects multi-host deployments sharing one server.

The send queue is unbounded. If SynchroServer is unreachable, metrics accumulate in memory without limit while the sender retries.

Exports are never pruned. `ExportToFile` rotates daily but nothing removes old files. On one host this reached 6.3 GB. If you enable it, prune externally.