

StreamGate Linux User Guide

StreamGate 1.9.7 · Customer delivery document

StreamGate is a camera ingest and distribution gateway for private deployments. Linux packages are portable [tar.gz](#) archives with optional systemd lifecycle scripts.

This guide applies to StreamGate 1.9.7. An offline backup always includes the data directory and may also include selected tasks' sealed recording directories. Stop Gateway, Core, and other recording writers before running [streamgate backup](#); copying and verifying more recordings extends downtime. After restoring, check recording paths, indexes, and retention before releasing task protection and enabling recording. The recording timeline links only saved events to complete segments with verified UTC sidecars.

Earlier upgrades: the one-time 1.9.6 settings migration

Stop the old version and retain its data directory. Use the installer to upgrade an installed deployment. For portable packages, replace program files while preserving [data](#), licenses, and recordings.

On the first upgrade from a version earlier than 1.9.6, StreamGate backs up the configuration database and restores system media policy and each channel's on-demand ingest, playback output, and transcoding settings to the 1.9.6 defaults. On-demand ingest is off by default. Devices, channels, and main/substream input sources are retained, as are network and CORS settings, accounts, API tokens, recording configuration, and license identity. Back up the existing data directory before upgrading.

This adjustment runs once. Later 1.9.6 restarts and future releases do not repeat it. The original configuration backup is stored under [data/backups](#). After startup, select any required playback protocols, transcoding, and substream output policies in System Settings and the channel editor.

Package contents

- [streamgate](#): Gateway and Web Console entry point.
- [core/streamgate-core](#): local media runtime.
- [lib/](#): packaged runtime libraries required by [streamgate-core](#).
- [sidecars/mediamtx/](#): bundled media distribution service.
- [configs/](#): default configuration files.
- [runtime/ca/](#): public CA trust bundle used to validate HTTPS, RTMPS, and RTSPS inputs.
- [licenses/](#): license and public-key files.
- [使用说明.pdf](#): Simplified Chinese guide for the current platform.
- [docs/zh-TW/使用說明.pdf](#): Traditional Chinese guide for the current platform.
- [docs/en-US/StreamGate-User-Guide.pdf](#): this English guide.

- Professional and Enterprise packages additionally contain the three localized Public API PDFs and [docs/streamgate-public-api-v1.openapi.yaml](#); Standard packages contain no API document.
- [install.sh](#) and [uninstall.sh](#): install or remove the systemd service and managed application files.
- [start.sh](#), [stop.sh](#), and [status.sh](#): service lifecycle helpers.
- [upgrade.sh](#): replace managed program files while preserving data, logs, licenses, and configuration.

The Web Console is embedded in [streamgate](#); no separate web directory is required. Standard and Professional packages use this same guide. Commands below use x86_64 Standard as an example; use the architecture and edition suffix from the actual archive. The HTML API reference and Postman collection are optional integration aids and are not included in runtime packages.

Analysis Node is an optional component distributed and installed separately. It is not embedded in the StreamGate package. A deployment that only uses ingest, distribution, recording, playback, and the Public API does not need Analysis Node. For intelligent analysis, install StreamGate first, then install Analysis Node 1.3.x for the same architecture and choose local attachment or remote-node mode. CUDA/OpenVINO accelerator packages and [sgcap](#) capability packages are also separate artifacts.

Run without installing

In a desktop session, the launcher can start or reuse the same-version portable instance and open the console:

```
./streamgate
./streamgate stop-local
```

[stop-local](#) accepts only a portable instance owned by the current account and login session with the matching data-directory lock. It does not stop systemd or terminate processes by name. When a custom data directory or listener is used, pass the same [-data-dir](#) and [-http](#) values to the stop command.

For foreground terminal operation, use:

```
tar -xzf streamgate-<version>-linux-x86_64-standard.tar.gz
cd streamgate-<version>-linux-x86_64-standard
./streamgate gateway
```

Web Console addresses:

```
On the StreamGate server: http://127.0.0.1:18080/
From another computer:  http://<server-lan-ip>:18080/
```

Install as a system service

```
sudo ./install.sh
```

Default locations:

- Program files: [/opt/streamgate](#)
- Configuration: [/etc/streamgate](#)
- Data and licenses: [/var/lib/streamgate](#)
- Logs: [/var/log/streamgate](#)

- Service: [streamgate.service](#)

Paths can be overridden during the initial installation:

```
sudo STREAMGATE_INSTALL_DIR=/data/apps/streamgate \  
STREAMGATE_DATA_DIR=/data/streamgate \  
./install.sh
```

Install a system service into persistent local directories. Do not use `/tmp`, `/var/tmp`, a network share, or a linked directory for managed program files.

Lifecycle scripts

Run the scripts from an independently extracted package directory.

Script	Privilege	Purpose
install.sh	root	Copies program files, creates the service account when needed, creates data/log/configuration directories, and registers and starts systemd.
start.sh	root	Starts only the service bound to the current installation identity. It does not fall back to an unmanaged foreground process.
stop.sh	root	Stops only the bound service and never terminates processes by global name.
status.sh	installation owner; root for a system install	Validates the installation marker and actual service binding and emits JSON status.
upgrade.sh	root	Replaces managed leaves whose ownership hashes match, preserving a transaction backup and the original enabled/running state.
uninstall.sh	root	Removes only unchanged StreamGate-owned program leaves and the bound unit. Preserves configuration, data, licenses, logs, modified files, unknown files, and separately managed Analysis Node files.

Typical workflow:

```
tar -xzf streamgate- <version>-linux-x86_64-standard.tar.gz  
cd streamgate- <version>-linux-x86_64-standard  
sudo ./install.sh  
sudo ./status.sh
```

Custom service identity and ports:

```
sudo STREAMGATE_SERVICE_NAME=streamgate-site-a \  
STREAMGATE_SERVICE_USER=streamgate-site-a \  
STREAMGATE_INSTALL_DIR=/opt/streamgate-site-a \  
STREAMGATE_CONFIG_DIR=/etc/streamgate-site-a \  
STREAMGATE_DATA_DIR=/var/lib/streamgate-site-a \  
STREAMGATE_LOG_DIR=/var/log/streamgate-site-a \  
STREAMGATE_HTTP_ADDR=0.0.0.0:19080 \  
STREAMGATE_CORE_CMD_URL=http://127.0.0.1:19082/cmd \  
./install.sh
```

The default listener is **0.0.0.0:19080**. Each side-by-side service must use a unique loopback Core command port. The first installation may also set the release manifest URL:

```
sudo STREAMGATE_HTTP_ADDR=0.0.0.0:19080 \  
STREAMGATE_CORE_CMD_URL=http://127.0.0.1:19082/cmd \  
STREAMGATE_RELEASE_MANIFEST_URL=https://hbrun.com/releases/streamgate/latest.json \  
./install.sh
```

Reinstall and upgrade preserve established environment values. To change them later, update the service environment file and restart the service:

```
sudo sh -c 'printf "%s\n" \  
"STREAMGATE_HTTP_ADDR=0.0.0.0:19080" \  
"STREAMGATE_CORE_CMD_URL=http://127.0.0.1:19082/cmd" \  
"STREAMGATE_RELEASE_MANIFEST_URL=https://hbrun.com/releases/streamgate/latest.json" \  
> /etc/streamgate/streamgate.env' \  
sudo systemctl restart streamgate
```

TLS media inputs always validate the certificate chain and host name. If an enterprise deployment uses a private CA, add this line to the environment file before restarting:

```
STREAMGATE_SOURCE_TLS_CA_FILE=/etc/streamgate/private-source-ca.pem
```

The CA file must be a regular, non-linked PEM file owned by the effective installation owner, with no group or other write permissions.

Camera ingest and playback settings

After adding an ONVIF device or RTSP channel, check the input codec and resolution. The channel main stream uses the selected input, which may be a camera substream; it is not necessarily the camera's physical main stream. Stream & Output has separate Main stream and optional Substream settings. Without a substream, only the main stream is available.

In System Settings > Media & Playback, Console playback stream sets `media.defaultViewingQuality`: `auto/high/fluent` mean Auto, Main stream, and Substream. This applies to console preview, details, and Standard/Professional video walls. Auto selects by window size and presentation; an unavailable substream falls back to main. The channel UI no longer has a default-playback selector. Legacy `viewing.defaultQuality` remains only for Public API playback-URLs requests without `quality`; explicit `quality=high/fluent/auto` still overrides it. The system console choice does not change Public API behavior, recording sources, or GB28181 delivery.

The main stream offers Use system, Source stream, Compatibility, and Always transcode. Substream sources are Device substream Profile, Device substream URL, and Transcode from main input. A device substream also has its own four output modes and parameters. Changing one stream does not overwrite the other. Save changes to apply them; they do not alter the camera's encoding.

Transcode from main input offers only Compatibility and Always transcode. A new configuration defaults to Always transcode, preserving source dimensions/frame rate and inheriting the system preset; explicitly set substream parameters when lower bandwidth is needed. It shares the original input with the main stream and encodes independently; it does not re-encode the main stream's encoded output. It can reduce gateway-to-viewer traffic, not camera-to-gateway ingest traffic.

- Compatibility converts only H.265 to H.264 and passes H.264 through. Always transcode continually encodes. Both allow independent bitrate, dimensions, frame-rate and GOP settings; the encoder backend is global. Compatibility ignores these parameters when conversion is unnecessary; it does not transcode H.264 merely to resize it. Transcoding uses CPU/GPU resources; check Actual Output for the active codec and backend.
- Transcoding Keyframe interval (ms) is specified in milliseconds; an empty field or 0 inherits the global interval (default one second). Shorter intervals can increase bitrate or storage. Passthrough does not change camera GOP; configure source GOP on the camera or NVR.
- On-demand ingest enabled means the source is pulled only while viewing, recording, analysis, or another task needs it, and can stop after becoming idle. Cold first frames may take longer. On-demand ingest disabled keeps the source connected while the channel is enabled. Active recording or analysis can keep the source connected even with on-demand ingest enabled.
- LL-HLS pre-generation under System Settings > Media & Playback offers Off, Selected channels, and All always-on channels. It is off by default and applies only to enabled, always-ingested channels. Continuous packaging uses CPU and memory caches, not disk by default; on-demand channels remain on demand.

For faster first frames, disable on-demand ingest, use browser-decodable H.264/AAC, and evaluate a roughly 1-2 second source keyframe interval on the camera. To reduce camera uplink traffic, enable on-demand ingest, lower device bitrate, or use a device substream. Viewing the main stream and device substream together may open two camera inputs. If only one lower-bitrate stream is needed, use the device substream as the channel input.

In System Settings > Media & Playback, choose Save bandwidth, Balanced or Prefer quality. The adjacent gear opens Transcode settings for each preset's quality strength and optional bitrate limit, plus the shared keyframe interval. Select the global encoder backend under Advanced playback & resources below.

Automatic presets preserve source resolution and frame rate. Set dimensions or a frame-rate limit explicitly in the channel's main/substream output when needed. Main and substream Use system independently inherit the global transcoding mode and preset; enabling transcoding does not create an unconfigured substream.

Preset	Default quality strength	x264 CRF	Intel QSV QVBR quality
Save bandwidth economy	4	28	28
Balanced balanced	6	25	25
Prefer quality quality	8	22	22

Higher quality strength preserves more detail and usually needs more bitrate. Strength is not a native encoder value: x264 uses CRF+VBV, QSV uses QVBR when supported, and NVENC/AMF use their

supported quality-priority VBR modes. OpenH264 and other backends without the corresponding mode fall back to a bitrate budget, not equivalent CRF quality. Backend values do not imply equal quality or recovery of detail already lost at the source.

The three `economy/balanced/quality` presets are stored in `media.videoTranscodeParameters.automaticPolicy`. Each `qualityLevel` is an integer `1..10`; zero/omission uses the built-in value, with nondecreasing preset order. Optional `maxBitrateKbps` accepts zero or an integer `256..64000` kbps. Zero/empty lets the system estimate a budget. A limit is neither a fixed bitrate nor a hard network cap including audio, packaging and retransmission. Unsupported quality modes are reported as `bitrate-fallback` in actual output.

An omitted, empty or all-zero channel `video.automaticPolicy` inherits the whole system policy; a nonzero object replaces it, using built-in defaults for missing fields. A positive channel `bitrateKbps` selects manual bitrate and bypasses automatic presets. All transcoding outputs use the global encoder backend; legacy channel `encoderPreference` remains readable but does not override it. A zero channel keyframe interval inherits the global value, whose default is 1000 ms.

Compatible mode passes an H.264 device stream through and transcodes H.265 to H.264. Transcode from main input explicitly creates an independently encoded output even when the main source is H.264. These source types differ; the generated substream does not re-encode the main output's encoded data.

Legacy `sourcePercent/referenceBitrateKbps/referenceCeilingKbps`, `automaticBitrate` and `hevcBitrateReservePercent` remain accepted for internal budgeting/fallback, not quality strength. Budgets consider measured source bitrate and actual output dimensions/frame rate; before measurement they use a format estimate. Legacy `maxWidth/maxHeight/maxFps` inside automatic presets no longer change the picture. Explicit channel `video` geometry and frame rate remain effective.

Cancel leaves the page unchanged; Apply edits the draft, and Save system settings makes it effective. Applying without edits creates no unsaved state. Shared-encoding edits preserve a legacy fixed bitrate; editing the automatic policy or choosing Use automatic bitrate switches to automatic mode. Read runtime state for the adopted backend, rate-control mode and native quality value. No Public API system-settings endpoint is added.

Saving a global preset or policy may briefly reconnect active video outputs that use the changed value. Independent channel parameters, explicit policies, and manual bitrates are not overwritten; unused preset changes do not rebuild the current output. Codec or dimension changes recalculate targets. Significant source frame-rate changes also update the automatic estimate after independent stable windows and a cooldown, without requiring a large source-bitrate change; small frame-rate jitter does not trigger adjustment. Short VBR fluctuations do not trigger second-by-second changes; sustained large changes require stable windows and a cooldown before adoption at a keyframe boundary. Reconnect a device that does not report its new codec. Public API players must still handle stream endings, track changes, and stalled playback.

In older configurations without a separate substream output policy, device Profile/URL substreams share the main stream's output policy. Generated substreams use the old output policy while the main stream uses the system policy. Viewing does not rewrite the configuration. The first editor save creates independent policies while preserving the previous behavior of both streams. An explicit Use system selection for the device substream makes it inherit the system independently.

In the Channel tab, set Recording stream to Main or Substream, then choose Original data or Transcoded data under Recording data. Main means the channel's main stream, not necessarily the camera's physical main stream. Substream is unavailable when not configured. An independent device substream can record its original data or its output; a generated substream has no separate original input and displays and uses only its output. Transcoded data follows the selected stream's actual output. When transcoding is unnecessary, that output is saved directly and the choice is retained.

New tasks default to the main stream's original data. Older tasks without a recording-stream selection use Main. Opening a task does not rewrite its settings; System processing (existing setting) retains its previous behavior until a new recording target is explicitly chosen. If changing the substream source or disabling it invalidates a target, choose a valid target or turn recording off before saving.

The Channel and Recordings pages manage the same task; selecting a recording target does not change its switch or schedule. Turning recording off preserves the schedule; turning it back on resumes it. Manage schedules, segments and retention in Recordings. Deleting a device retains its channels. Deleting a channel removes recording tasks and indexes, not disk files; they disappear from the playback list. To pause recording, use the switch.

GB28181 upstream cascading

Set Stream mode (all cascaded channels) under GB28181 > Upstream cascade. Single stream uses Main by default, or Substream for all channels. Dual stream provides both channel streams to the upstream platform. The policy applies to all locally published channels; there are no per-channel overrides. A channel without a substream uses Main without blocking cascading. Main means the channel's main stream. Substream can use a device substream or be transcoded from the main input. Each stream uses its own saved output parameters.

Saved policy is displayed separately from draft selections. Active sessions determine the actual stream in use. Older configurations without a stream mode display Keep existing stream handling. Opening the page or saving other fields retains that behavior. A new policy is adopted only after explicitly choosing and saving a stream mode; the upstream platform then needs to request playback again.

Upgrade and transaction safety

For a directly launched portable instance, run `./streamgate stop-local` and confirm that the old process has exited before starting the new release. If another version still owns the management port, the new launcher reports the version that is actually running instead of opening the old console as though it were the new release.

Extract the new package into a new directory, then run:

```
sudo ./upgrade.sh
```

Upgrade preserves `/etc/streamgate`, `/var/lib/streamgate`, and `/var/log/streamgate`. Every lifecycle command must receive the same custom paths, service name, and service user. Never use the installed program directory itself as the package input.

The six lifecycle entry points require Python 3 and share [gateway-lifecycle.py](#). The packaged [node-lifecycle.py](#) is a transaction-safety library only; it does not contain Analysis Node executables, models, or capabilities.

Before changing the host, the scripts validate absolute paths, the real systemd [FragmentPath](#), [ExecStart](#), and configuration binding. Directory links, escaping links, unknown services, and unexpected hard links are rejected. Managed program directories and every ancestor up to / must be non-linked directories owned by root or the effective installer and must not be group/other writable (022). This rule does not require variable data and log directories owned by the service account to be changed to root.

Transaction backups are placed next to the program directory as [.streamgate-transaction-*](#). A failed operation restores owned files and the original service state only when identity and leaf hashes still match. Concurrent changes or unmanaged child processes cause a fail-closed result and require review. Do not delete the transaction directory before the result is understood.

Analysis Node capability installation, activation, permissions, and ownership are managed by the independent Node transaction. StreamGate lifecycle scripts do not take over or delete that directory. If an upgrade finds a legacy bundled Node without independent ownership, it stops and requires the independent Node package to adopt or upgrade it first.

Migration from an official 1.8.2 installation

Public 1.8.2 installations without [STREAMGATE_INSTALL_MANIFEST.json](#) require explicit legacy-archive evidence. The helper accepts only the four official Gateway 1.8.2 archives whose SHA-256 values are registered in the release. The archive must match the installed x86_64/aarch64 architecture and Standard/Professional edition. Keep the historical archive outside the installed program root, state directories, and newly extracted package.

First perform the non-root archive check:

```
LEGACY_GATEWAY=/srv/streamgate-release-evidence/streamgate-1.8.2-linux-x86_64-standard.tar.gz
python3 -B ./gateway-lifecycle.py legacy-info --legacy-archive "$LEGACY_GATEWAY"
```

[legacy-info](#) validates archive evidence only. It does not prove an installed instance and does not perform an upgrade. If a local Analysis Node 1.2.2 is present, also supply the independently pinned official Node archive:

```
LEGACY_NODE=/srv/streamgate-release-evidence/streamgate-analysis-node-1.2.2-linux-x86_64.tar.gz
python3 -B ./gateway-lifecycle.py legacy-info \
  --legacy-archive "$LEGACY_GATEWAY" \
  --legacy-node-archive "$LEGACY_NODE"
```

After separate authorization for the root operation, run preflight and upgrade. Add the same [--legacy-node-archive](#) argument when a legacy local Node exists. Custom installations must pass the same [--install](#), [--config](#), [--data](#), [--log](#), [--service](#), and [--user](#) values on every command.

```
sudo python3 -B ./gateway-lifecycle.py preflight --legacy-archive "$LEGACY_GATEWAY"
sudo ./upgrade.sh --legacy-archive "$LEGACY_GATEWAY"
```

Preflight is read-only but requires the installation owner's authority to inspect root-owned state.

Upgrade verifies the official unit, real service binding, configuration paths, installed program bytes, and permissions. Modified programs, unrelated services, links, and hard links remain rejected. Do not run or remove the old installer to make migration pass.

A successful migration returns a [backup](#) path containing a private [legacy-rollback.json](#) receipt and per-file backups. Retain that exact path, the new package and helper, and the legacy archive. To request rollback of the most recent legacy migration:

```
TRANSACTION='/opt/streamgate-transaction-<returned-id>'
sudo python3 -B ./gateway-lifecycle.py rollback \
  --transaction "$TRANSACTION" \
  --legacy-archive "$LEGACY_GATEWAY"
```

Replace the placeholder with the returned backup path. Rollback requires all managed leaves and the receipt to remain unchanged and to represent the latest migration. A reinstall or later upgrade invalidates an older rollback receipt. Configuration, database, recordings, licenses, keys, capabilities, providers, models, and unknown files remain untouched.

Rollback does not downgrade the database schema. Before upgrading, make a separate cold backup of configuration and data; a newer database may not open in an older version.

Logs and diagnostics

Use `journalctl -u streamgate` and the Web Console log page for routine troubleshooting. Diagnostic output identifies versions, component startup, listeners, and errors without intentionally exporting full credentials. Apply a site-specific retention policy to logs and exported diagnostics. Never include active license files, API tokens, camera credentials, or full private source URLs in public evidence.

Licensing

The public Standard and Professional runtime packages both provide a long-term free baseline of five input channels when no license is installed. Professional capability evaluation requires a signed trial license. Evaluation video has no watermark; term, capacity, and feature scope follow the issued license. Paid capacity, Professional features, and commercial rights require a matching license.

Standard capabilities include GB28181 registration and keepalive, catalog publishing, live media requests, and media sessions; advanced workflows require the applicable Professional or Enterprise entitlement. Public REST API, WebHook, and MQTT are unavailable in Free and Standard. A valid Professional trial, Professional, or Enterprise software license enables them, subject to the [openapi](#) entitlement and runtime package. REST access and REST integration management also require the corresponding Token scopes; event receivers use WebHook signatures or MQTT credentials separately. Analysis licensing does not replace these software entitlements. Professional trials are only for permitted non-production evaluation.

The license page shows deployment identity and enabled features. Analysis Node and each analysis capability require their own entitlements.