The Small-Agency GTFS Publishing Playbook

TransitFare • 2026-08-23

The Small-Agency GTFS Publishing Playbook

A single, end-to-end reference for a small transit agency taking its schedule from a spreadsheet to a live feed that riders see inside the apps they already use. It follows the whole lifecycle in seven stages: deciding to publish, building a clean static feed, validating before go-live, getting distributed everywhere, going realtime, adding fares and alerts, and keeping the feed alive after launch.

Every stage is a one-page summary of what the work involves and the decisions it forces. Each one points to a deeper article on transitfare.com for the mechanics, so this document stays a map rather than a manual. Read it start to finish once, then keep it beside whoever owns the feed. Two printable tools sit at the end: a pre-publish checklist and a distribution-targets table.

The General Transit Feed Specification (GTFS) is an open, vendor-neutral standard. Nothing in this playbook requires a particular tool, and most of it can be done by hand by a determined planner. It is written for the agency running 10 to 50 buses, where one or two people own the schedule, the feed, and everything else. Where a platform genuinely saves that person time, the playbook says so and shows TransitFare Cloud as one way to run the lifecycle, not the only one.

How to Use This Playbook

Work the stages in order the first time. Publishing is sequential: a feed cannot be distributed before it validates, and it cannot go realtime before there is a static feed for the realtime data to reference. After the first launch the work becomes a cycle, not a line, and Stage 7 is where most of an agency’s ongoing effort actually lives.

Treat the two references at the end as living documents. Print the pre-publish checklist and run it before every release, not just the first. Keep the distribution-targets table current as the list of places riders look for transit data keeps growing.

Stage 1 • Deciding to Publish

Before any files get built, an agency decides who owns the feed and how it will be produced. There are three honest routes: build and maintain GTFS in-house, hire a consultant to produce it, or have it generated as part of a platform that already holds the schedule. Each has a real cost, and the cheapest option on day one is often not the cheapest across the years a feed has to be maintained.

The decision that outlasts all three is data ownership. Whoever produces the feed, the agency should own the underlying schedule data, the public feed URL, and the accounts that

register the feed with Google, Apple, and the data catalogs. A feed that lives only inside a consultant's login becomes a problem the day that relationship ends.

Publishing is also increasingly not optional. State guidelines, such as California's minimum GTFS guidelines, and some grant deliverables tie funding to a published, valid feed. Confirm the exact requirement that applies against the current program document rather than assuming last year's rule still holds.

For the full comparison of the three routes and their true long-run costs, see GTFS: in-house or consultant?. For the ownership questions to settle in writing before work starts, see who owns your GTFS data?.

Stage 2 • Building a Clean Static Feed

The static feed, GTFS Schedule, is the foundation everything else references. It is a set of comma-separated text files zipped together. A minimum viable feed needs agency.txt, stops.txt, routes.txt, trips.txt, stoptimes.txt, and a service calendar in calendar.txt or calendardates.txt. Everything after that adds precision.

Four files carry most of the difficulty. Stops.txt has to place every stop at accurate coordinates, because a stop dropped in the wrong block is the error riders notice first. Shapes.txt draws the path a vehicle actually follows, so the line on the map matches the road rather than cutting between stops. The calendar files describe which days each service runs, including the exception days, holidays and reduced-service days, that a plain weekday-weekend pattern misses. Stoptimes.txt has to keep times in strict order along each trip.

Complex service is where agencies wrongly assume they need outside help. Interlined blocks, where one vehicle runs several routes in sequence, timed transfers at a pulse hub, and frequency-based turn-up-and-go corridors are all expressible in standard GTFS. They are harder to hand-build, not impossible, and a good producer handles them natively.

For the file-by-file mechanics, see getting stops.txt right, why route geometry matters, and service calendars without gaps. To have the feed generated from scheduling data so a service change updates it as a byproduct, see automating GTFS from your scheduling software. The full field reference is what is GTFS?.

Stage 3 • Validating and Versioning Before Go-Live

A feed that opens without an error in a spreadsheet can still fail the moment a trip planner ingests it. Validation is the discipline of catching those failures before riders do. The canonical MobilityData GTFS validator is the reference check: it separates hard errors, which block acceptance, from warnings, which degrade quality without blocking. The goal before go-live is zero errors and a deliberate, documented decision about each remaining warning.

Versioning is the other half of go-live discipline. Every published feed carries a service date range, and a new version should never overwrite a live one until it has been validated on its own. A simple sign-off step, one person builds, a second person validates and approves, prevents the most common cause of a broken public feed: a well-meant edit pushed live on a Friday afternoon.

For the specific errors that most often block acceptance and how to clear them, see the GTFS errors that break your feed. For what feed quality scores actually measure beyond pass or fail, see what feed quality scores mean. Validate against the official rule set documented at gtfs.org rather than a tool's private interpretation of it.

Stage 4 • Getting Seen Everywhere

A valid feed sitting on a hard drive helps no one. Distribution is the stage that puts the feed in front of riders, and it starts with one thing everything else depends on: a stable, public, permanent URL that serves the current zip and does not change. Accidental URL changes, usually from a website redesign, are a common cause of a feed silently going stale in the apps. Host the feed where its address is managed as infrastructure, not as a page someone might move.

From that stable URL, the feed is submitted to and catalogued by several destinations, each with its own path. Google and Apple are direct submissions the agency controls. The open catalogs, the Mobility Database and Transitland, make the feed discoverable to the wider ecosystem of transit apps at once, so an agency does not have to approach each app individually. The table below lists the main targets and how each one works.

Distribution Targets

- Target | What it is | How the feed gets there | Notes
- Google Maps | The most-used consumer trip planner | Register the feed URL in the Google Transit Partner Dashboard; Google reviews before it goes live | Review can take time on a first submission; a clean, valid feed clears review faster
- Apple Maps | Apple's transit layer on iPhone and Mac | Contact Apple's transit data team directly | Apple runs its own review and may request corrections
- Mobility Database | The community-run successor catalog to the old TransitFeeds registry | Add or claim the feed at mobilitydatabase.org | Makes the feed discoverable to many apps at once; keep the entry pointing at the stable URL
- Transitland | An open aggregation of transit feeds worldwide | Register the feed URL with Transitland | Feeds the many apps and tools that consume Transitland's aggregation
- Transit apps | Rider apps such as the Transit app, Moovit, and Citymapper | Most ingest from the open catalogs above; some accept direct submissions | Catalog listing usually reaches these without contacting each app

The Google submission and review process is covered step by step in how to publish GTFS data to Google Maps. For the Apple path and the broader app ecosystem, see trip planners beyond Google. Why the feed URL must stay stable, and how to keep it up, is the subject of feed hosting and uptime. Register only with the official catalogs named above; the current onboarding details live at mobilitydatabase.org and gtfs.org.

Stage 5 • Going Realtime

A static feed tells riders when a bus is scheduled. GTFS-Realtime (GTFS-RT) tells them where it is now and whether it is running late. It is a separate feed, in a different format, and it references the static feed by the trip and stop identifiers already published in Stage 2. That linkage is why the static feed comes first: a realtime feed pointing at trip IDs that do not exist in the current schedule produces nothing riders can use.

GTFS-RT carries three kinds of information in three feeds: vehicle positions, trip updates with predicted arrival times, and service alerts. Positions and predictions come from onboard location data, the Computer-Aided Dispatch / Automatic Vehicle Location (CAD/AVL) system reporting where each vehicle is. The bar that matters is freshness and reliability. A realtime feed that lags or stalls is worse than no realtime feed, because riders learn to distrust it and stop checking.

For how the three realtime feeds fit together, see GTFS-Realtime, explained. For what happens when a feed stops updating and how to catch it, see when a GTFS-Realtime feed goes stale. The specification for each message type is documented at gtfs.org.

Stage 6 • Fares and Alerts

Two optional layers turn a schedule feed into something closer to a complete rider experience. GTFS-Fares v2 expresses what a trip costs: the fare products, the rules for who pays what, and the transfers between them. It is expressive enough to model most small-agency fare structures, from flat fares to zone and transfer rules, so a rider sees the price in the same app that shows the trip. It is intricate to hand-build, which is the honest reason many agencies delay it.

Service alerts are the rider-facing half of GTFS-RT: the detour, the cancelled trip, the stop moved for construction. Alerts are published as part of the realtime data and are the fastest way to reach riders inside the apps they already have open, rather than only on the agency’s own website. An alert that names the affected route and stop, with a clear start and end time, is the one riders act on.

For fares, see GTFS-Fares v2 for small agencies. For the mechanics of a realtime service alert, see what is a GTFS-Realtime service alert?, and for getting alerts to actually surface in the apps, see how to publish transit alerts to Google Maps.

Stage 7 • Keeping the Feed Alive

The first publish is the smallest part of the work. A feed is a living document that has to change every time the service does, and a stale feed is arguably worse than none, because riders act on schedules that no longer run. Most ongoing effort is a cadence: a regular schedule-change workflow that updates the feed, validates it, and republishes on the same stable URL without a gap.

Two recurring events test that cadence. A planned service change, a new pick or a seasonal shakeup, needs the new version validated and staged so it goes live exactly when the service does, not before and not after. A detour or disruption needs a fast path through realtime alerts, because a printed schedule change cannot keep up with a road closure that starts tomorrow. An agency that has rehearsed both handles them as routine rather than emergencies.

For a service change that does not break the feed, see a schedule change without breaking the feed. For handling detours and disruptions in near real time, the same alert mechanics from Stage 6 apply. Set a calendar reminder tied to each service pick so a feed update is never something a rider notices before the agency does.

Pre-Publish Checklist

Run every item before a feed, or a new version of one, goes live. A blank box on release day is how a broken feed reaches the apps.

- [ ] All required files present: agency.txt, stops.txt, routes.txt, trips.txt, stoptimes.txt, and a service calendar.
- [ ] Every stop placed at accurate coordinates inside the actual service area.
- [ ] Shapes drawn so the map line follows the road, not a straight cut between stops.
- [ ] Calendar covers every service day, including holidays and exception days, with no gaps in the active date range.
- [ ] Stop times in strict order along each trip, with no impossible timings.
- [ ] Complex cases handled where they apply: interlined blocks, timed transfers, and frequency-based service.
- [ ] Feed passes the MobilityData validator with zero errors and a documented decision on each remaining warning.
- [ ] Service date range set correctly, and the new version validated on its own before it replaces the live one.
- [ ] A second person has reviewed and signed off, separate from whoever built the feed.
- [ ] The public feed URL is stable, permanent, and serves the current zip.
- [ ] Distribution accounts confirmed and owned by the agency: Google, Apple, Mobility Database, Transitland.

- [ ] If publishing realtime, the GTFS-RT feed references trip and stop IDs that exist in the current static feed.
- [ ] A maintenance cadence is set, tied to each service pick, so the next update is scheduled, not improvised.

Where TransitFare Fits

TransitFare is a transit technology company founded in 2012 that builds GTFS publishing tools for small and mid-sized transit agencies running 10 to 50 buses. The seven stages in this playbook are vendor-neutral, and an agency can run every one of them by hand or with a consultant.

Where TransitFare Cloud helps is in collapsing the lifecycle into one system. The static feed is generated from the schedule the agency already keeps, so a service change in Stage 7 updates the feed as a byproduct rather than a separate task someone has to remember. Validation and versioning happen in the same place the feed is built. The public URL is managed as infrastructure, so the Stage 4 hosting problem does not recur. GTFS-RT vehicle positions and trip updates are produced from CAD/AVL, and realtime alerts from Transit Alerts on the same platform. Fare files are not part of the generated feed today. The platform has a strong production reliability record and has processed 60 million+ fare validations, and over 90% of customers expand to additional modules.

To see how the whole lifecycle runs on one platform, see GTFS publishing in TransitFare Cloud. For a lighter entry point if this is a first feed, start with getting started with GTFS.

This playbook is freely reusable with attribution to TransitFare, transitfare.com. It carries no cost and no form to download.