Smart pi ECOsystem



Smart Public Information for the smart city.

Smart pi takes information from across the city and delivers it to the people who need it: as an announcement, on a display, through a help point or straight to a hearing aid, the moment it matters.

One ecosystem, designed to look, feel and work as one

Every product works on its own, and better together.

IN

Where information comes from

RTA · MIC · PBX · SPL · Fire panel

SMART PI CORE

Decides what to say, where and when

DVA · DVA-GTFS · EYE · DSP · SIP · FSM · LOG · NMS

OUT

How it reaches people: hear, see, ask

NAM · NAM-HP · EDC · EYE-HD · HLP

Smart pi

Realtime data

Filter, merge and shape service feeds for the information systems that depend on them.

The feeds you need.
The data your systems use.

Aggregate and prepare transport information before it reaches the passenger-information applications.

Smart pi RTA brings timetable and realtime feeds into a node-based processing system. Filter by agency, combine sources and remove data that downstream applications do not need.

Give Smart pi EYE and DVA GTFS the relevant service information without making each application manage every upstream feed and its processing rules.

Several sources

Bring timetable and realtime feeds together.

Selected agencies

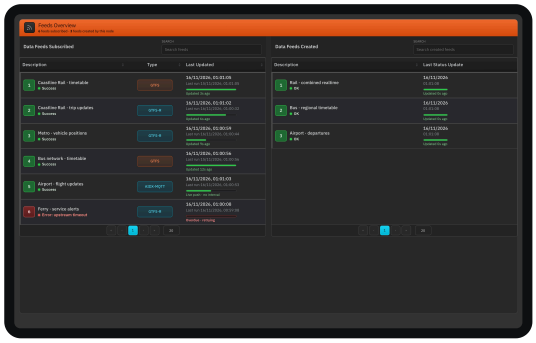
Keep the operators needed for the destination.

Useful outputs

Trim the payload around its intended use.

Visible operations

Review feeds, jobs, nodes and logs.



Inputs and outputs together

Subscribed source feeds beside created outputs.

GTFS

Timetable and static transport data.

GTFS-R

Realtime service information.

Jobs

Source, agency and output rules.

Nodes

Capacity and redundancy choices.

SmartPi UI preview with fictional sample feeds and illustrative operational data.

From external feeds to useful service information

Keep source acquisition and output processing in one managed chain.

Sources
Timetable and realtime

→

Ingester
Subscribe and ingest

→

Rules
Merge, filter and trim

→

Outputs
Prepared feed data

→

Applications
Displays and announcements

- Processing jobs select their source feeds and agencies, set their own interval and define their output folder.

Separate source management

The ingester handles external feeds. Processing jobs select and transform the data for each downstream use.

Reuse the prepared result

Create the outputs needed by the information system, so downstream applications can focus on displaying or announcing the service.

Keep the decision visible

- Which source feeds are assigned to the job?
- Which agencies and realtime entity types are included?
- What is removed from the output?
- How often does the job run, and where is its result written?

Part of the wider ecosystem

Smart pi EYE
Passenger information displays.

Smart pi DVA-GTFS
Service-driven announcements.

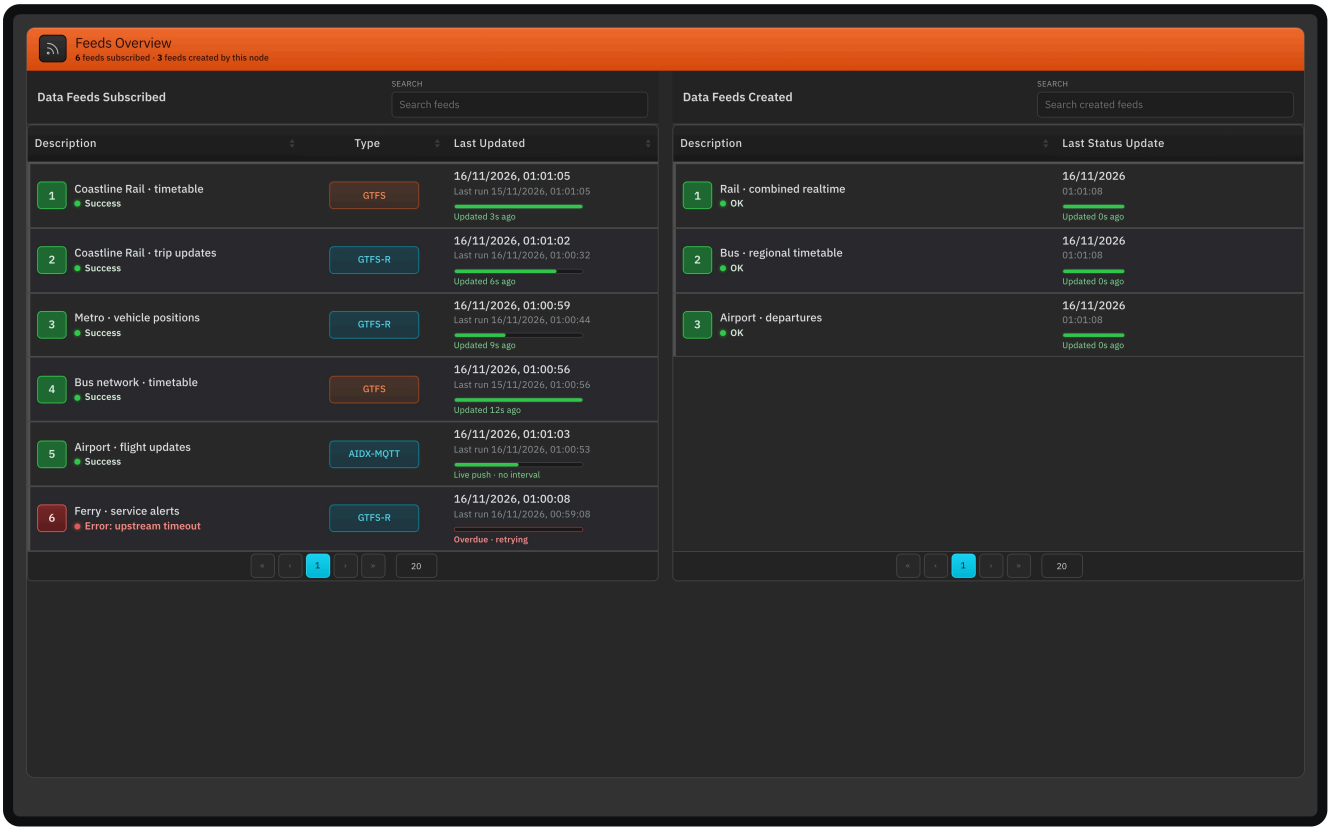
Smart pi DVA
Announcement playback.

Smart pi NMS
Wider device monitoring.

ⓘ Illustrative processing workflow. Application destinations are configured for the installation.

What arrived?
What was created?

Bring incoming source health and prepared output status into the same review.



Subscribed feeds and created feeds
Review source type, status, last update and the outputs created by the node.

Follow the source freshness

A subscribed feed can be healthy, late or failing. Compare its update timing with the configured schedule and source behaviour.

Check the created output

A source update and a created feed are separate stages. Confirm the expected output is present and current before investigating the consumer.

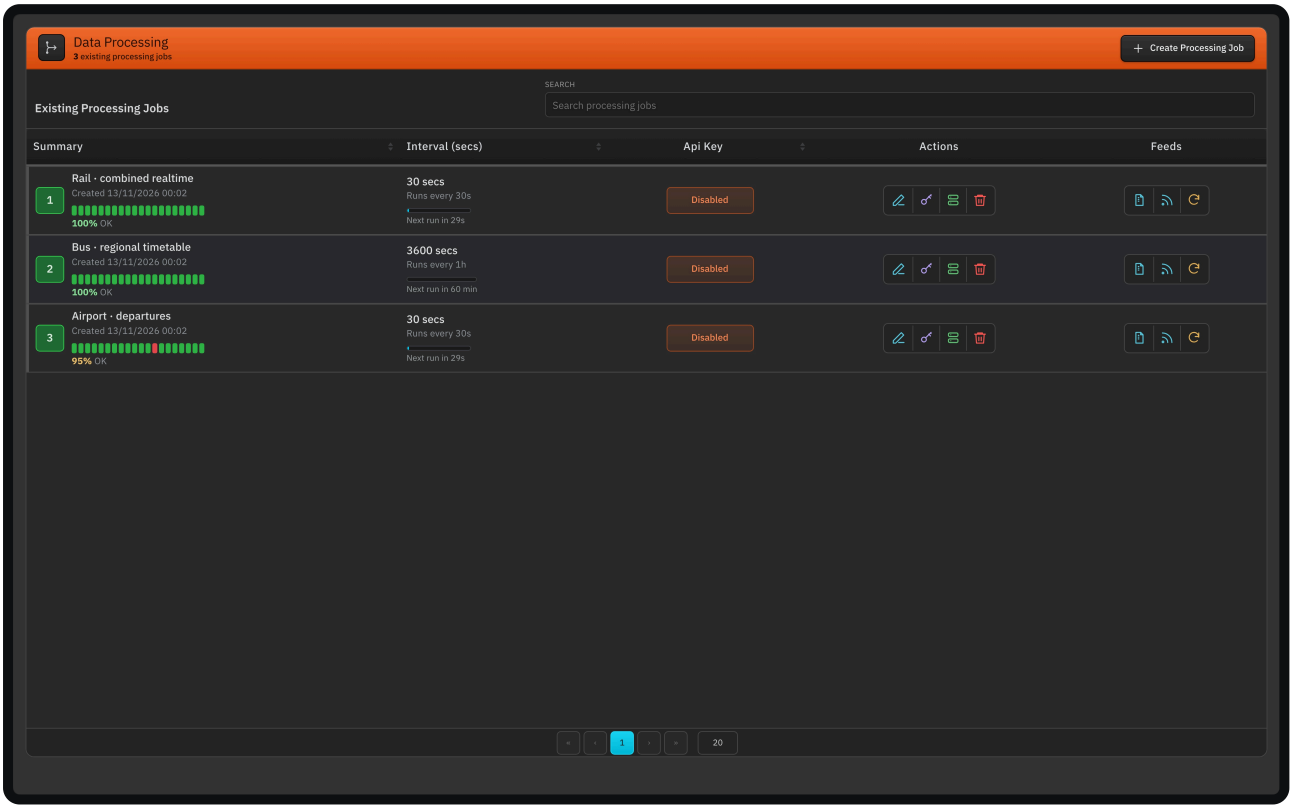
Start at the point that changed

If a service update is missing downstream, trace it from the source feed through its processing job to the created output. Use the corresponding log scope to narrow the investigation.

SmartPi UI preview with fictional sample feeds and illustrative operational data. One upstream timeout is included to illustrate a source fault.

A clear purpose.
A repeatable output.

Define what the job creates, how often it runs and where the result belongs.



Processing jobs
Job identity, interval and status in the native processing list.



A job's identity and schedule
The sample job has its own summary, processing interval and output folder.

A job for each output need

Choose the input set and policy for the destination. Different jobs can serve different agency groups or output requirements.

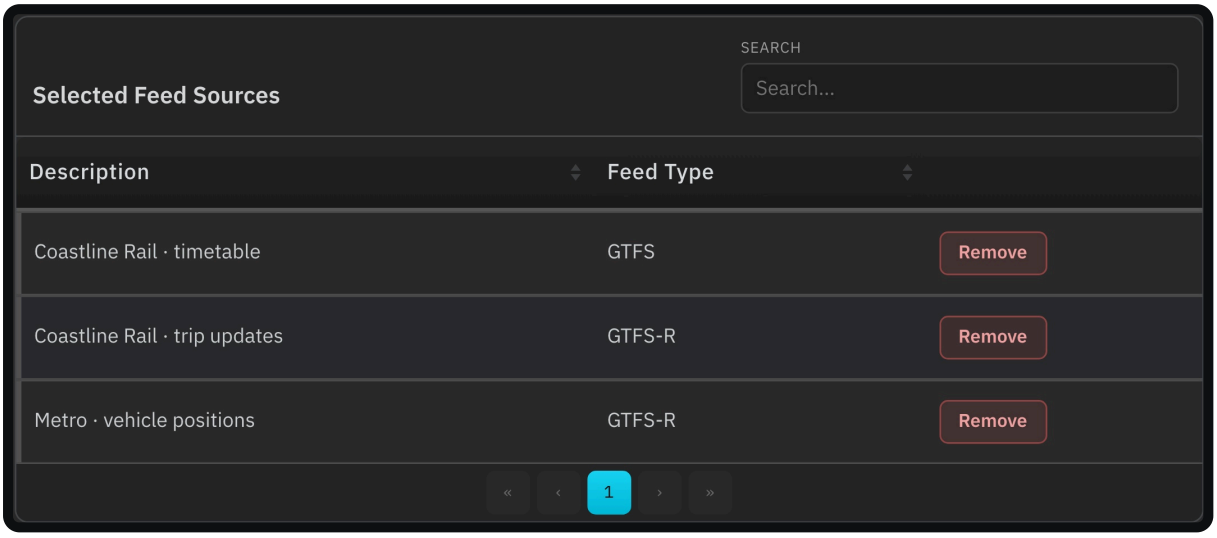
An interval for the data

Set the processing interval around the source and the downstream use. A realtime update stream and a timetable refresh need different scheduling.

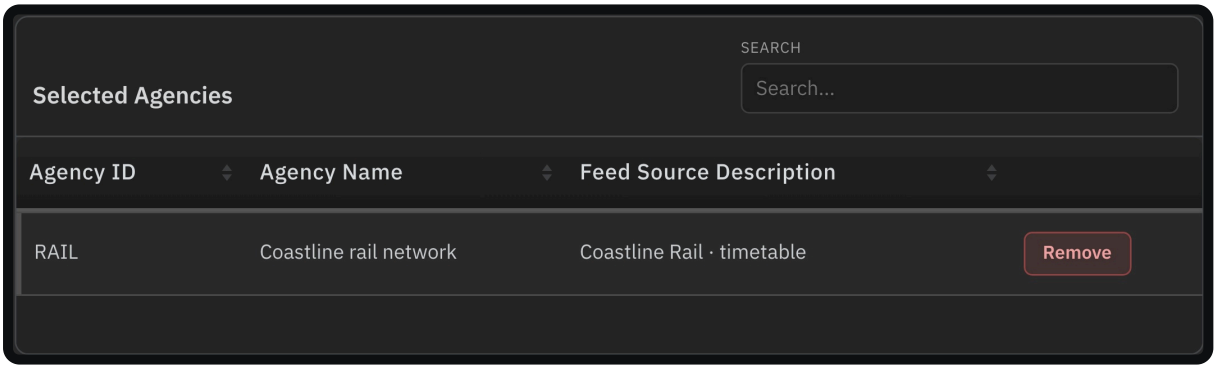
SmartPi UI preview with fictional sample feeds and illustrative operational data.

Combine the sources. Keep the agencies you need.

Use the job's rule editor to make the output boundary explicit.



Selected source feeds
A sample job combines timetable, trip-update and vehicle-position sources.



Selected agencies
This example includes the fictional Coastline rail agency.

Merge the relevant feeds

Bring several sources into the job when the output needs a combined data set. Keep the selected list understandable to the operator.

Filter by agency

Choose the agencies needed for the output, rather than passing every available operator to the consuming system.

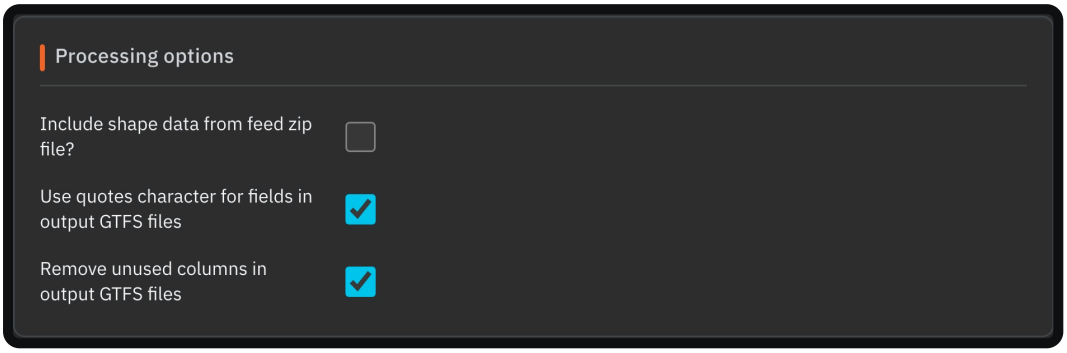
A practical example

A display group needs one rail agency from a wider transport feed. Select the relevant timetable and realtime sources, add that agency and verify the prepared output against representative services.

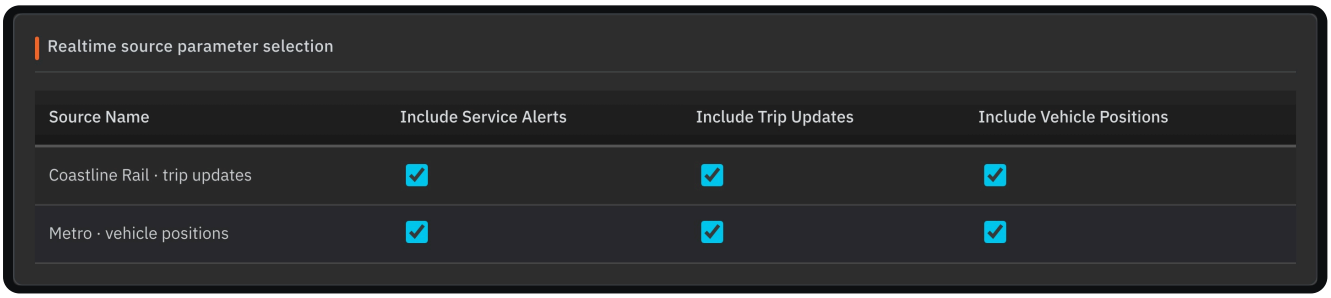
SmartPi UI preview with fictional sample feeds and illustrative operational data. Coastline Rail is fictional and does not identify a client deployment.

Less unused data. The right realtime detail.

Shape the payload around the application that will consume it.



Static output options
Shape data, quoted fields and unused-column removal are set on the job.



Realtime entity selection
Choose service alerts, trip updates and vehicle positions for each realtime source.

Remove unnecessary bulk

Exclude shape data and trim unused columns when the destination does not need them. Preserve the fields required by the consuming application.

Choose the realtime content

The job can select alerts, trip updates and vehicle positions per source. Align that selection with the downstream information task.

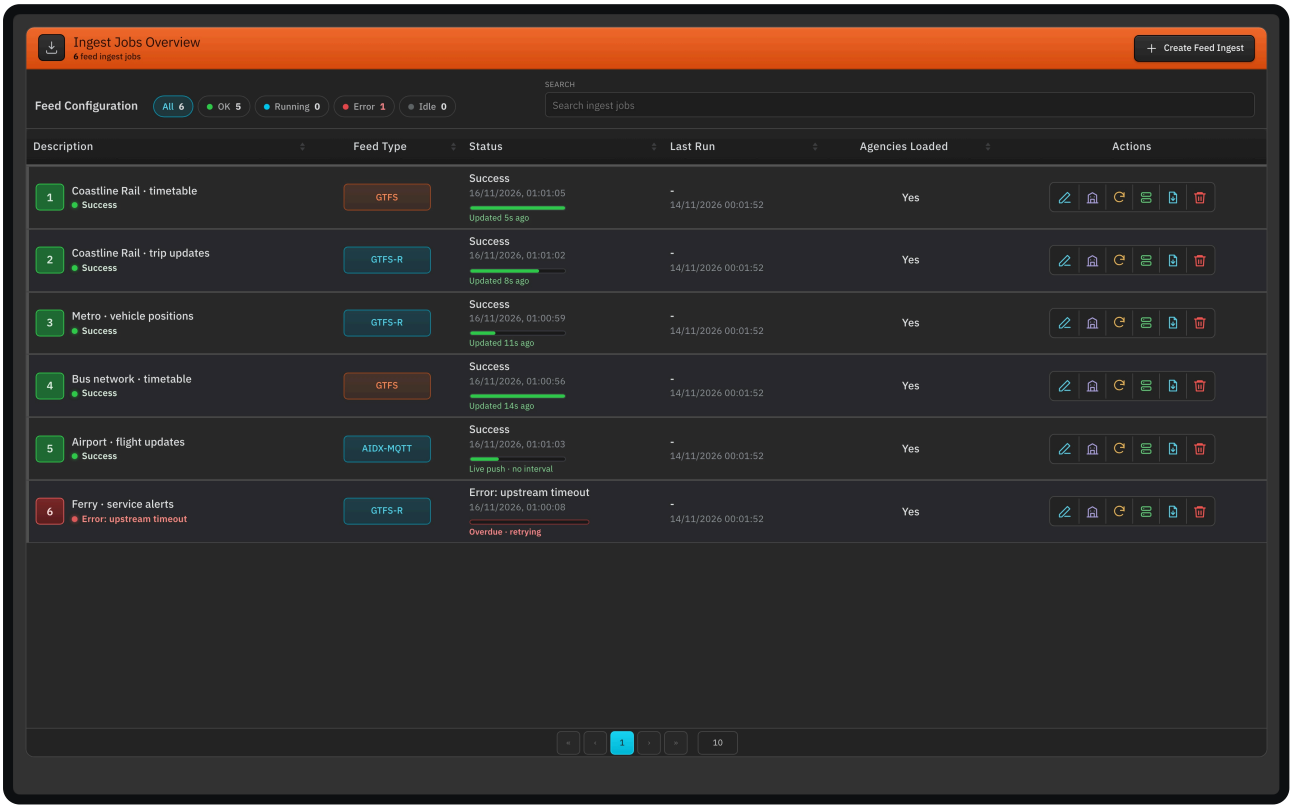
Validate the consumer as well as the file

Check that the prepared feed still carries the service details the display or announcement application expects, including changes and exceptional service conditions.

SmartPi UI preview with fictional sample feeds and illustrative operational data. Rules configure sample data; the browser preview does not run a live GTFS conversion.

Bring different sources into a consistent workflow

Use source adapters and node assignments around the feeds the system needs.



Source ingestion
Configure and review external source jobs and their assigned nodes.

Standard transport data

RTA supports GTFS and GTFS-R sources for modes such as rail, bus, light rail and ferry services.

Other source protocols

Plugins map custom protocols to GTFS / GTFS-R outputs. Published options include a SIRI module and AIDX feeds over MQTT or REST.

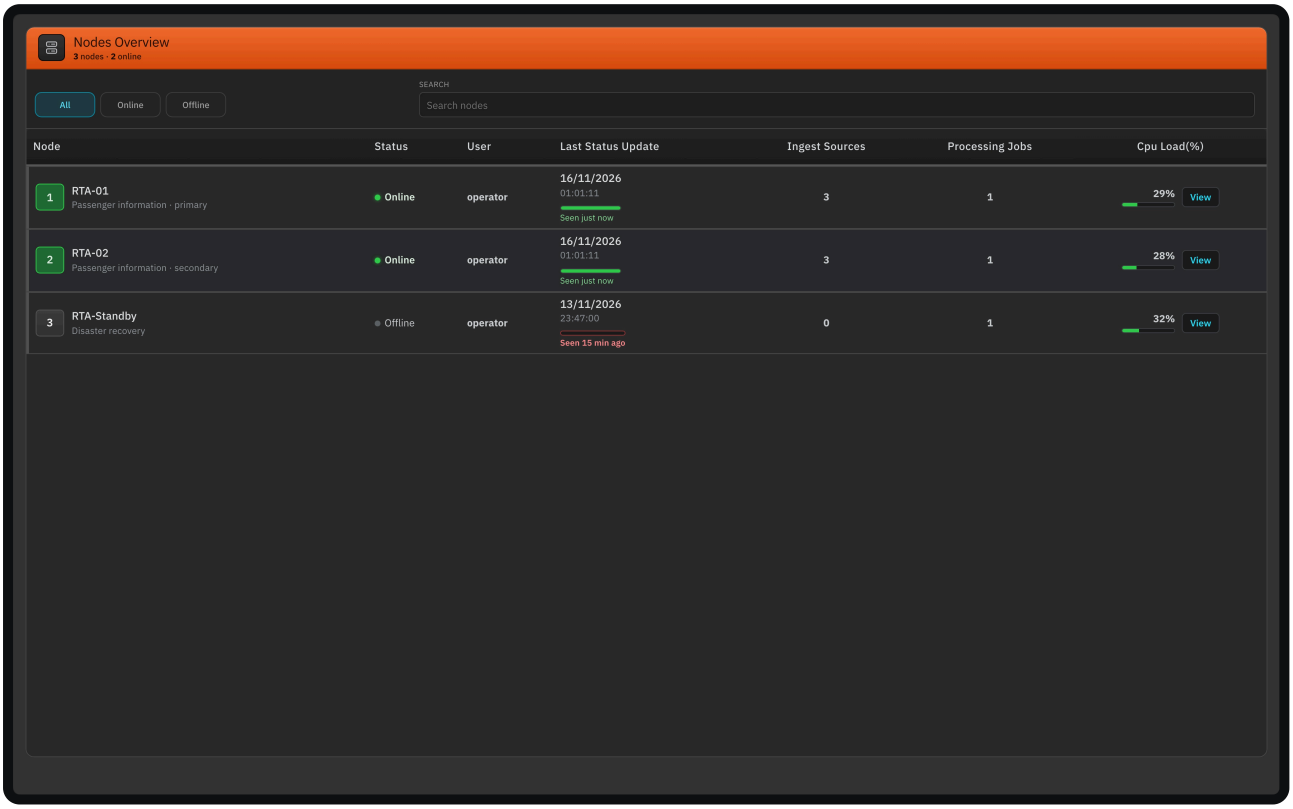
Keep source ownership clear

Record where each feed comes from, how it updates and which processing jobs depend on it. Assign the ingestion work to the appropriate nodes and review the result before changing the consumers.

SmartPi UI preview with fictional sample feeds and illustrative operational data.

Grow the processing system. Keep its behaviour visible.

Use nodes for capacity, redundancy or both, around the actual feed workload.



Node status and workload
Review configured nodes and their operational state.

Capacity where it is needed

Add nodes as the feed and processing workload grows. Select node assignments around the installation's jobs.

Redundancy by design

The published architecture supports redundant deployments. Plan the complete node and database arrangement, then test the intended failover behaviour.

One shared data model

RTA uses a MariaDB database shared by the nodes. Its node-based design supports on-premises, cloud and hybrid deployment arrangements.

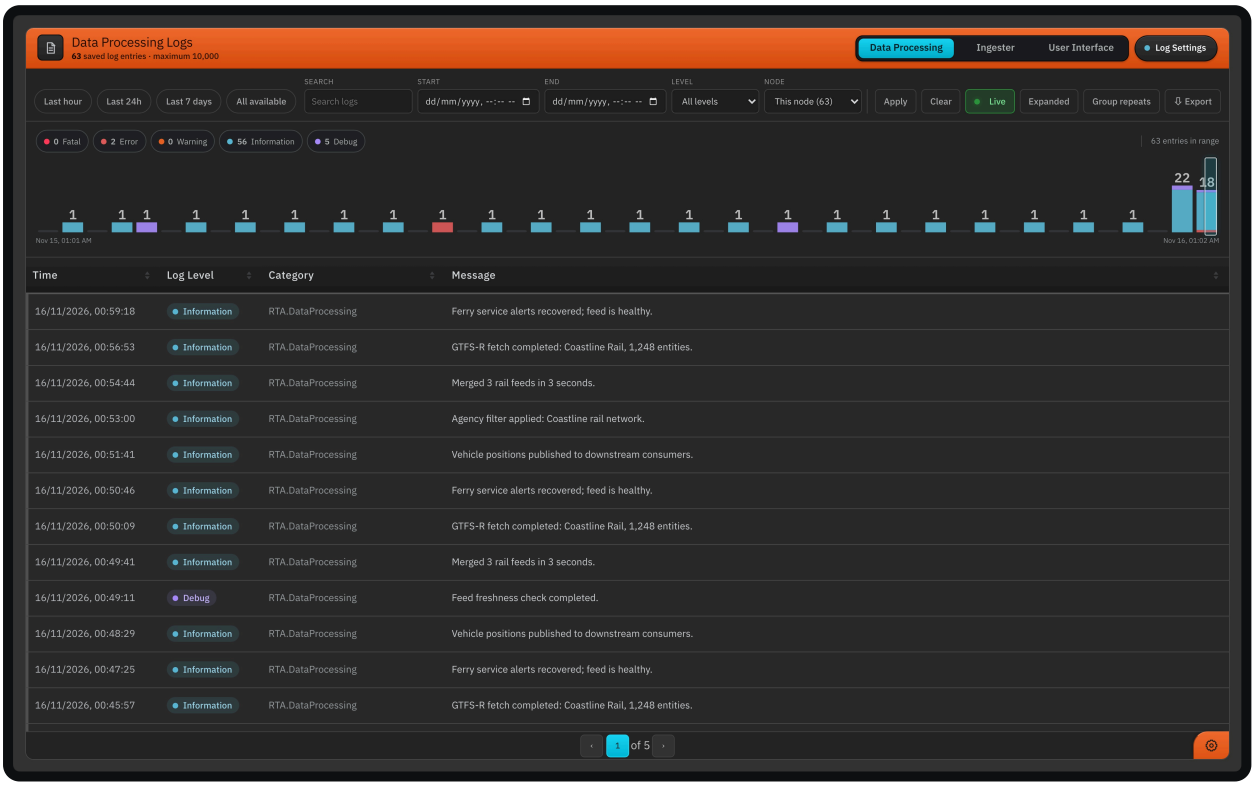
Prove the output through a change

Include a node or service interruption in the acceptance plan. Check which work continues, how recovery is indicated and whether the downstream feed remains suitable for its consumer.

SmartPi UI preview with fictional sample feeds and illustrative operational data.

Which stage changed? Who changed the setup?

Operational logs and identity-based audit records support investigation and change review.



Processing events over time
Select levels and a period, then inspect the entries behind the timeline.

Separate the log scopes

Ingester, Data Processing and User Interface logs help locate the stage involved in a problem. Filter by severity, dates and relevant text.

Trace configuration changes

The published audit capability records changes against the signed-in LDAP user. Keep identity and permissions aligned with operational responsibility.

Review timing with status

Compare the feed update, job timing and log entries around the same interval. Retention and automatic cleanup settings should support the review process.

SmartPi UI preview with fictional sample feeds and illustrative operational data. The timeline and entries are illustrative operational data.

Define the feeds. Prove the prepared result.

Build the configuration around the applications that depend on the data.

1. **List the sources**
   Identify timetable, realtime and any optional protocol adapters, with their ownership and update behaviour.

2. **Design the outputs**
   Define the agency groups, source combinations and downstream applications for each prepared feed.

3. **Set the jobs**
   Assign sources and agencies, processing intervals, output folders and payload options.

4. **Plan the deployment**
   Choose nodes, database arrangement, network access and LDAP groups; include the required redundancy.

5. **Test the full chain**
   Verify source changes, job results, consumer behaviour, fault indication and recovery, then retain the accepted configuration.

Published starting specification

2 CPU cores
Starting processor allocation.

4 GB memory
Starting memory allocation.

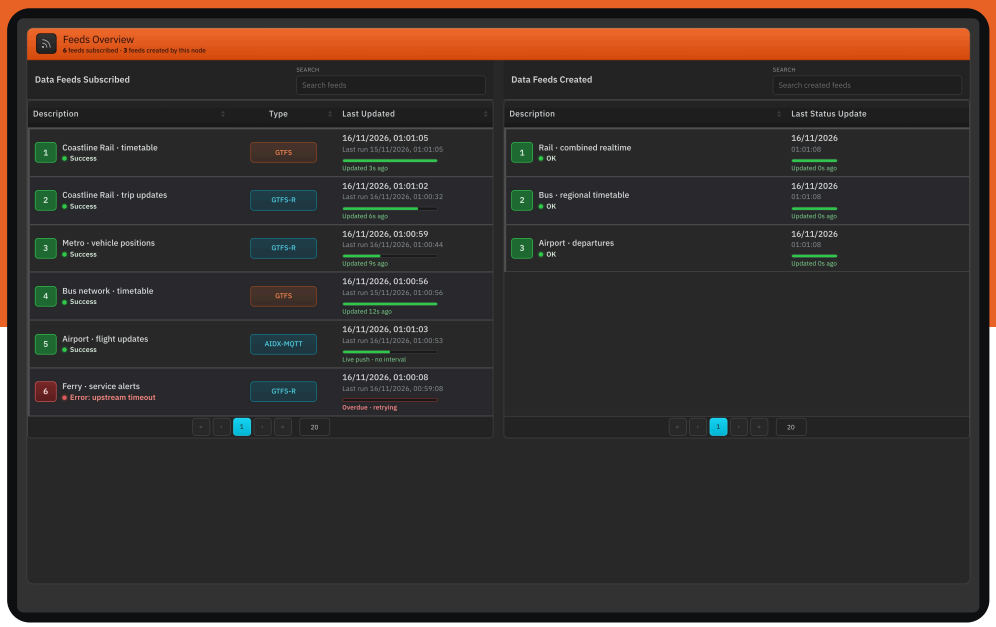
50 GB / feed
Starting storage allowance.

Windows Server or Ubuntu LTS. Runs as a service, with browser access over HTTPS. Confirm resources for the feed size, job count, processing intervals and retention.

SMART PI RTA

Prepare the data. Deliver the information.

Talk to our team about your feeds, agencies, processing rules and downstream systems.



Web
www.smart-pi.info

Email
info@smart-pi.info

Sales in Australia and New Zealand: Qvest
+61 2 8011 0500

Scan to visit smart-pi.info

Smart pi ECOsystem

Smart pi is a T&M Media Pty Ltd brand · ABN 67 612 111 369
Native interface with fictional sample feeds. Specifications may change without notice.