

Vector Scheduling Third-Party Integrations

Reducing Duplicate Data Entry Across
Mission-Critical Systems: Use Cases, Software
Examples & API Implementation Reference



Table of Contents

Purpose & How to Use This Guide.....	02
API Fundamentals.....	03
Integration Patterns at a Glance.....	04
Use Case Deep Dives.....	05
A. Payroll & Time-and-Attendance Export	
B. Records Management (RMS) & Incident Reporting Roster Sync	
C. Station Dashboards & Situational Displays	
D. Legacy & Custom Payroll/HR Builds	
E. Third-Party Managed Connectors	
Endpoint Cheat Sheet.....	09

Purpose & How to Use This Guide

This document is a directory of the ways customers connect Vector Scheduling to other software. For each use case, it names the software Vector customers use, explains how the integration is typically built, and gives a plain-English summary of how it would be done against the Vector Scheduling API (formerly the CrewSense API).

A note on payroll: Vector Scheduling has no native integration with any payroll system. Every payroll-related row in this guide is either a manual/scheduled CSV export or a custom build against the API — there is no out-of-the-box connector to buy or turn on.

API Fundamentals

Everything in this guide is built on the same underlying RESTful API, so the mechanics are worth covering once up front.

Getting credentials

Only users with the “System Settings” permission can generate API credentials: Admin → System Settings → API / Integrations → Generate New API Credentials. This produces a Client ID and Secret Key. Anyone holding these has full API access to the account, so they should be treated like a password and only shared with a trusted integrator.

Base URLs

- United States: <https://api.crewsense.com/v1>
- Canada: <https://api.canada.crewsense.com/v1>

Authentication (OAuth 2.0, client credentials grant)

Exchange the Client ID / Secret for a bearer token, then attach that token to every subsequent call. Tokens are valid for one week; after that, request a new one the same way.

```
curl -v https://api.crewsense.com/oauth/access_token \
  -d "client_id=YOUR_CLIENT_ID" \
  -d "client_secret=YOUR_SECRET_KEY" \
  -d "grant_type=client_credentials"

# → { "access_token": "...", "token_type": "bearer",
#      "expires": 1426274440, "expires_in": 86400 }

curl -v https://api.crewsense.com/v1/schedule \
  -H "Authorization: Bearer <access_token>"
```

Full endpoint-level documentation lives at the Vector Scheduling Support Center (API topic) and developer.crewsense.com. This guide summarizes the pieces relevant to each integration pattern below rather than reproducing the whole reference.

Integration Patterns at a Glance

Every third-party integration we have on record falls into one of six patterns:

Pattern	Typical Software	Data Direction	Who Builds It
A. Payroll / Time & Attendance Export	ADP, Paylocity, Paycom, UKG, Oracle HCM, BIAS, Black Mountain, MIP, JD Edwards	Scheduling → Payroll	Customer (CSV) or customer/3rd-party dev (API)
B. Records Mgmt (RMS) Roster Sync	ESO, Emergency Reporting, ImageTrend, EPR Fireworks, Zoll	Scheduling → RMS	Vendor-enabled or 3rd-party managed
C. Station Dashboards	First Arriving, FirstWatch, Chief360	Scheduling → Display	Vendor-guided, vendor-built
D. Legacy / Custom Builds	AS-400, homegrown apps	Scheduling ← ► Legacy system	Customer IT / contracted dev
E. Third-Party Managed Connector	ESO (via MCP), Oracle PeopleSoft (via MCP)	Scheduling ← ► Target system	Paid third-party integrator
F. HR / Personnel System Push	Workday, UKG Pro, BambooHR, Paycor, NEOGOV	HR → Scheduling	Customer IT / vendor or contracted dev (API)

Use Case Deep Dives

A. Payroll & Time-and-Attendance Export

What it solves: Getting worked hours, overtime, and approved leave out of Scheduling and into whatever system actually runs payroll.

Software Examples on Record

Vendor / Product	Method
ADP	CSV
Paylocity	CSV / FTP
Paycom	CSV
Kronos / UKG Dimensions	CSV export only (limited/mixed)
Oracle Cloud HCM	CSV (HCM Data Loader)
Oracle PeopleSoft Payroll	API (custom, via Mission Critical Partners)
Oracle SuccessFactors	CSV
BIAS (Springbrook)	CSV
Black Mountain	CSV
Abila (now MIP)	CSV
JD Edwards	CSV
BS&A	API

How it's done with our API

The API exposes a purpose-built endpoint for this: GET /payroll.

```
curl -v https://api.crewsense.com/v1/payroll -G \  
  -H "Authorization: Bearer <token>" \  
  -d "start=2026-08-01 00:00:00" \  
  -d "end=2026-08-15 00:00:00"
```

Each item in the response is one worked shift, time-off entry, callback, or misc.-hours entry, already carrying the employee, work type/subtype, work code, assignment, labels, notes, and total_hours — essentially a pay-period-ready payroll feed. GET /payroll/{user_id} scopes the same call to a single employee, which is useful for spot-checks or a per-employee reconciliation report.

If a payroll system needs finer-grained detail than the /payroll feed provides (e.g., break time, qualifiers on a shift, or a specific work subtype), build the same feed manually from GET /shifts (worked shifts), GET /time_offs (approved leave), GET /users (employee/role data), and GET /work_types (pay code definitions), then join them client-side before exporting.

Use Case Deep Dives

B. Records Management (RMS) & Incident Reporting Roster Sync

What it solves: Getting an accurate “who was on duty” roster and assignment record into the RMS/incident-reporting system automatically, instead of an admin re-keying it for every NFIRS-type report.

Software Examples on Record

Vendor / Product	Integration Type
ESO	No native path — built and managed by Mission Critical Partners (3rd party, charges a fee) using the API
Emergency Reporting	Native — must be enabled by Emergency Reporting on their side, then turned on in Vector Admin settings
ImageTrend	Native — ImageTrend builds it and sends Vector what's needed to enable it; ImageTrend charges
EPR Fireworks	Managed entirely by Fireworks — Vector-side step is just an Admin creating an API key for them
Zoll Data Systems (Zoll Fire Reports)	API — Vector-side step is an Admin creating an API key

How it’s done with our API

The workhorse endpoint here is GET /schedule, which returns a full day-by-day roster for a date range in one call — assignments, the shifts (and the employees) filling them, qualifiers, labels, time off, callbacks, and trades, all nested under each date.

```
curl -v https://api.crewsense.com/v1/schedule -G \  
  -H "Authorization: Bearer <token>" \  
  -d "start=2026-08-17 08:00:00" \  
  -d "end=2026-08-18 08:00:00"
```

A nightly (or per-shift-change) job typically calls /schedule for the relevant window and pushes the resulting roster into the RMS’s crew/incident model — this is exactly the pattern Mission Critical Partners runs for ESO, and what EPR Fireworks and ImageTrend run on their own side once they have an API key.

- Use GET /shifts?user_id= for a single employee’s assignment history instead of the whole roster.
- Use GET /qualifiers and GET /users/{user_id}/qualifiers to carry certifications/roles (e.g., Captain, Firefighter) into the RMS’s crew roster fields.
- Use GET /users/{user_id}/certifications if the target system also wants certification records rather than just qualifiers.

Use Case Deep Dives

C. Station Dashboards & Situational Displays

What it solves: A large screen at the station or HQ showing who's on shift today, tomorrow, and yesterday, alongside training/equipment status — or the same “who's on duty” picture pushed to mobile devices for dispatch and station alerting.

How it's typically built

Nothing native. The customer generates an API key/secret in Admin, hands it to the dashboard vendor, and the vendor's own team builds and maintains the read-only pull — the customer just follows the vendor's setup guide. Chief360's proposal-stage connector follows the third-party-managed pattern from Use Case E instead, since Mission Critical Partners is building it.

How it's done with our API

These integrations are read-heavy and current-day focused. GET /schedule (or GET /shifts) scoped to “today” returns who's on duty right now; GET /groups and GET /labels supply the “A Shift / B Shift” and unit color-coding these dashboards typically display. For mobile alerting specifically (Chief360's pattern), the same /schedule pull maps on-duty personnel to whatever roles the alerting system needs to notify.

Use Case Deep Dives

D. Legacy & Custom Payroll/HR Builds

What it solves: Agencies on an older mainframe-style payroll system, or a fully homegrown application, that can't consume a standard export format.

How it's done with our API

Same building blocks as Use Case A (GET /payroll, GET /shifts, GET /users, GET /time_offs), but the transformation logic is fully custom — built by the customer's own IT staff or a contracted developer to match whatever fixed-width or proprietary format the legacy system requires.

Use Case Deep Dives

E. Third-Party Managed Connectors

What it solves: When there's no native path and the customer doesn't want to build or maintain their own integration, a paid third-party integrator builds and operates the connection on the customer's behalf.

How it's done with our API

The integrator is handed a Vector API Client ID/Secret by the customer, authenticates the same OAuth2 client-credentials flow as any other consumer, and polls GET /schedule (or GET /shifts) on whatever cadence the target system needs. Vector doesn't build, host, or charge for this path — the integrator's fee is between them and the customer.

Use Case Deep Dives

F. HR System Pushing Into Scheduling

What it solves: The HR platform, not Scheduling, is the system of record for who's employed, what role they hold, and what leave they've earned. Rather than an admin manually re-keying every new hire, promotion, transfer, or accrual adjustment into Scheduling, agencies have HR push that data over on a set cadence — new user profiles, role/rank changes, and leave bank balances — so Scheduling stays in sync with HR without someone reconciling the two by hand.

How it's typically built

Nothing native. This is the one use case that runs the opposite direction of the rest — data comes into Scheduling instead of out. The customer generates an API key/secret in Admin, hands it to whoever owns the HR side of the integration (their own IT team or the HR platform's integrator), and that team builds and maintains the push. There's no packaged connector for this; it's either a scheduled import job or a custom build against our API, same honesty-first framing as the payroll use case.

How it's done with our API

These integrations are write-heavy, the reverse of most other use cases. POST/PUT /users handles new hires, role changes, and profile updates; PUT /time_offs (or the accrual-bank equivalent) reconciles leave balances so a vacation payout or FMLA adjustment made in HR reflects in Scheduling without manual entry. GET /groups and GET /labels may still be read first, to map the incoming HR role/departments fields to the correct Scheduling group or shift assignment before the write happens.

Endpoint Cheat Sheet

The handful of endpoints that cover nearly every integration above:

Endpoint	Method(s)	Returns	Best used for
/oauth/ access_token	POST	Bearer access token (1-week expiry)	Every integration — required first call
/schedule	GET	Full day-by-day roster: assignments, shifts, time off, callbacks, trades, misc.	RMS roster sync, station dashboards
/payroll	GET	Flattened, pay-period-ready hours per employee	Payroll / time & attendance export
/shifts	GET / POST / PATCH / DELETE	Individual one-off shift records	Fine-grained roster or payroll builds
/users	GET / PATCH / PUT	Employee roster & profile data	Enriching any export with names/roles
/time_offs	GET / POST	Approved & pending leave	Payroll leave hours; RMS availability
/qualifiers	GET	Certification/role tags (e.g., Captain)	RMS crew-role mapping
/work_types	GET	Pay code / work type definitions	Mapping hours to payroll pay codes
/labels	GET	Crew Scheduler display labels	Dashboard color- coding
/groups	GET	Shift/platoon groupings (e.g., A Shift)	Dashboard and RMS crew grouping

Talk to an expert to learn more about how Vector Scheduling can integrate with your existing mission-critical systems to strengthen readiness with greater efficiency.

Request a Demo

