

IDRuns User Guide

Learn how to monitor workflows, maintain definitions, and troubleshoot scheduled work.

This guide mirrors the in-product Help screen (**Help > User Guide**). Topics marked *Administration* are visible only to users whose role allows them.

Getting started

The fastest way to orient yourself and begin monitoring work.

- Use the navigation menu to move between Operations, Definitions, Administration, Reports, and Help. Administration lists only the screens your role can use.
- In the Classic layout you can minimize the navigation tree; it also minimizes itself on narrow screens, and your choice is remembered in this browser.
- Use the color-scheme picker to choose your preferred appearance. Your selection is remembered in this browser.
- Start in Job Activity to see what is waiting, running, completed, or failed.
- Select Online (View > Online Users in the Classic layout) to see who else is signed in and to read or send messages. A badge counts unread messages.
- Help > About this installation shows which version is running here - the server and web app builds, the database schema and when the server started. Copy build information puts it on the clipboard and Contact us opens an email to info@idruns.com with it filled in. Help > About us tells you who builds the product.
- The IDRuns Tutorial (docs/TUTORIAL.md, also TUTORIAL.pdf) walks through the common tasks step by step with screenshots: signing in, monitoring runs, creating and scheduling jobs, planning work with AI, orchestrations, calendars, and administration.

Choose your appearance

Change the colors, shapes, surfaces, and interaction style without affecting your data.

- Open the color-scheme picker in the top bar. Its three swatches show the colors of your current scheme.
- Classic uses the original compact, square-edged interface. Coastal is clean and restrained, with soft surfaces and a teal accent.
- Electric uses angular controls, technical grid details, sharp navigation, and offset shadows. Sunrise uses warm glass surfaces, floating navigation, pill controls, and soft cards.
- Lime Pop uses chunky borders, dotted details, tactile shadows, and playful angles. Berry uses rounded surfaces, rich gradients, circular accents, and deeper elevation.
- The three Mondrian schemes are inspired by the painter Piet Mondrian: Mondrian Lines uses a flat black grid on an off-white canvas, Mondrian Bold adds red, yellow, and blue color-block chrome, and Mondrian

Full adds a complete color-block composition.

- Select any scheme to apply it immediately. The choice is saved only for you in the current browser and remains active after signing out or refreshing.
- The administrator-controlled Application layout is separate from your color scheme. It sets the navigation for everyone: Command Center (icon rail with a contextual header), Classic (top menu and navigation tree), or Sidebar (one full-height navigation panel). Your scheme works with every layout.
- On a narrow screen, the picker displays swatches without its text label. Select the swatches to open the same list of schemes.

Save and restore job-group versions

Keep named versions of a complete job group and compare or restore them later.

- Version History is an optional feature. An administrator turns it on under Administration > System Settings > Enable Version History; until then the menu items below are not available.
- In Definitions > Jobs, right-click a Job Group and choose Save Version. Save any pending form edits first. Enter a label and optional notes. Choose Version History on the same menu to browse saved versions.
- Version History compares the current group with a saved version, or two saved versions. Expand objects to see field and script differences.
- Preview Restore shows the exact group-tree changes before applying them. A recovery version is saved automatically, and existing scheduled runs keep their earlier configuration.
- Events and actions default to Use current. Choose Create local copy from archive for individual objects when you need archived settings without changing shared definitions.
- Existing variable values are always kept, including empty values. New variables use their archived initial values. These rules also apply to file imports.

How to create a job

Create a runnable definition, give it an owner and schedule, then verify it safely.

- Open Definitions > Jobs. Right-click Jobs in the left menu and choose New > Job, or open the relevant Job Group and use its New menu.
- To start from a plain-English description, choose Describe Your Job... on the + menu and type something like "Run backup.sh every weekday at 2am on DB01". Select Generate to see the proposed name, command, time window, repeat, suggested dependencies, and a confidence score. Under the proposal, Agent, Runtime user, and Calendar are dropdowns: the AI fills one in when your description makes it clear, and otherwise it says not specified and you choose from the list. Select OK to open it in the Job form for review; nothing is saved until you save there. If the work is really a pipeline (loops, conditions, approvals), select Build as an orchestration instead. This needs an AI API key configured by an administrator under Administration > AI Monitor, and is not available in the demo.
- To plan a whole piece of work, choose Plan with AI... on the + menu and say what you want to accomplish. The AI proposes a plan: a Job Group and the jobs inside it, with their agent, calendar, time window,

repeats, and the order they run in. Refine it by typing a follow-up (for example "run it at 3am instead" or "add a cleanup step at the end"), or edit names, commands, agents, and calendars directly and remove any job you do not want. Switch Build as to Orchestration (pipeline) to get one orchestration instead of jobs; when a request needs loops, conditions, or approvals, the AI suggests that switch. Select OK to create everything: the group, the jobs, and their dependencies (or the orchestration with its steps). If a step fails part-way, Undo moves what was created to the recycle bin. Check the commands first - the AI uses placeholders when it does not know your scripts. Nothing is created until you select OK.

- To organize related work first, choose New > Job Group. Enter a unique name, select its Workgroup owner, and configure the shared calendar, timezone, agent assignment, runtime user, variables, or time window that child jobs should inherit.
- After saving a Job Group, right-click it and choose New > Job to create a child job inside it. Groups can also contain nested Job Groups.
- On Program, enter a unique Name and the command to run. Choose FTP, Terminator, Approval Gate, Branch, or Plugin Job instead when the work needs one of those specialized job types (see Control runs, resources, and deadlines).
- Choose a Parent Group when the job should inherit settings. Every new job also needs an effective Workgroup owner, either selected directly or inherited from its parent. Type in the Filter groups box beside the Parent Group list to narrow it by name or job number; every word you type must match, and the group already chosen always stays listed.
- On Schedule, select a Calendar and timezone and set any effective dates, offsets, time window, or SLA deadline. A root job must also have an Agent or Agent List unless it is an in-process plugin job.
- Use Run for runtime identity, working directory, timeout, success criteria, and reruns, plus Priority and Resource Pool requirements. Add prerequisites on Dependencies and optional triggers on Job Events. Variables, Options, Run Book, and Notes hold the remaining settings.
- If an administrator has enabled Job Iteration (System Settings), the Job and Job Group forms also offer an Iterate over values section that fans one definition out across a list of values.
- A new Job or Job Group remembers your last-used Workgroup, and after saving it is selected and scrolled into view in the Jobs grid. A saved Job's History tab lists its runs across every production date, newest first; double-click a run for its details.
- Select OK to save. Enable the job when it is ready, then use Insert in Schedule for a controlled run and confirm the result in Operations > Job Activity.

Monitor job activity

Follow current and historical job runs and act on selected runs.

- Choose the production date with the date box or the Previous day and Next day buttons.
- Switch between the List, Flow, and Sequence views. The view you pick is remembered in this browser.
- Search or use a column filter to narrow the grid. Column headings remain visible while you scroll.
- Select checkboxes for bulk actions. Shift-click a second checkbox to select the range between them. Right-click a row for actions that apply to that run.

- Double-click a job group (or use its drill-in icon) to go into it and show just that group. Go Into Job does the same for a job; First, Previous, Next, and Last step through each time the scoped group fired.
- Your checked runs are remembered in this browser between visits.
- Review status, start and end times, duration, agent, and message details before retrying or changing a run. Open a run to see its status, output, history, dependencies, and audit log.
- Waiting runs show what they are waiting for: Waiting for Scheduled Start, Waiting for Time Window, Waiting for Rerun, Waiting on Resources (master capacity or a Resource Pool), Waiting on Iteration, Waiting on Group, or Deferred. The Reason on a run's Status tab lists each unmet prerequisite on its own line.
- On a run's Status tab, View / Edit Job Definition opens the job's full definition. It opens read only (the title says so) when you are not allowed to change it. Closing the form returns you to the same run in Job Activity.
- A run of an Approval Gate waits as Awaiting Approval. Right-click it (or several selected ones) and choose Approve or Reject. Only the gate's listed approvers and administrators can decide.
- Select the dependency-lines icon above the grid to draw a line from each job to the jobs it waits on. A dashed grey line means the dependency is in another Job Group. Inside Go Into Group, a dependency on a job outside the group is drawn as a short dashed line ending in a small circle; hover the circle to see the outside job's full path.
- The Sequence view compares expected dependency order with what actually ran. Each run is In order (started after all its prerequisites finished normally), Out of order (started before a prerequisite finished - check for an override, a changed definition, or a rerun), Needs review (cannot be judged automatically; open the run to check it), or Pending (not started). Show only problems lists just the Out of order and Needs review runs, with their count.
- Administrators also get a Recompile button, which rebuilds the day from current definitions, and a Clear button, which deletes the day's runs. After a day is cleared the daily compile does not rebuild it; use Recompile to bring its schedule back.

Control runs, resources, and deadlines

Pause for a person, choose a path by exit code, share limited resources, and track deadlines.

- Approval Gate is a job that never runs a command. Create it from New > Approval Gate in Definitions > Jobs. Once its dependencies are met it waits as Awaiting Approval. Approving it completes it normally and releases its dependents; rejecting it completes it abnormally. List approver usernames and/or approver workgroup IDs; administrators can always decide, and with no approvers listed only administrators can.
- Branch picks which downstream job runs next. Create it from New > Branch and add its upstream job on the Dependencies tab. It runs once that job finishes, whether it succeeded or failed. Add rules such as If exit code = 0, between 1 and 4, or >= 8, each with a target, plus an Otherwise rule. The first matching rule picks the target; every other target is Skipped.
- Resource Pools limit how many jobs use a shared resource at once, such as database connections or licenses. An administrator creates each pool under Administration > Resource Pools with a Max concurrent number of units.

- On a Job's Run tab (Priority & Resources), or a Job Group's Resources tab, add the pools and units the work needs. A group's requirement applies to every Job under it; a Job's own requirement for the same pool overrides the group's. A job that cannot get its units waits as Waiting on Resources until units free up.
- Priority (default 0) decides which job goes first when several are waiting on the same Resource Pool or on master capacity. Higher numbers are admitted first.
- SLA: on the Schedule tab of a Job or Job Group, set Finish within N minutes of starting, and/or a Finish by time of day (always in the timezone of the agent that runs the job). A warning alert is raised when 80% of the window has passed and a critical alert if the deadline is missed. Leave it blank to inherit the parent group's SLA.

React to events

Start follow-up work when a job changes state, a file appears, a variable changes, or the scheduler itself changes state.

- Open Definitions > Events. The tabs are Job Events, File Events, Variable Events and System Events; Job Trigger Types and System Event Triggers list the conditions each kind can react to. Event Activity (Operations) shows events as they fire.
- An event runs reusable Actions (email, SMS, alert, log, set a variable, or add a job to the schedule) in order. Create an Action once and reuse it from any event.
- **File Events** watch a folder on an agent. On the File Monitor tab choose the Agent, an optional Runtime User, the Directory and a File Mask, then choose what fires the event: a file arrives, is modified (rewritten in place after it fired and has settled again), disappears (deleted or moved away), or is recreated. A file must be stable for the configured time before it counts.
- On the Steps tab of a file event, build an ordered list: a file disposition (move, copy, rename or delete), copy or move to another agent, run a command on the agent, run an orchestration, wait, or an Action. Later steps see the file's new location. Use `<FilePath>`, `<FileName>` and `<FileSize>` in step fields. A step can run for only some triggers, and can stop the rest of the list when it fails.
- Use Browse next to a path field to pick a file or folder on the chosen agent's own disk instead of typing the path.
- Trigger now (on a saved file event) runs its steps for a file the monitor currently sees, or any full path you type, as if it had just arrived. Trigger History lists every firing and what each step did.
- **System Events** react to the scheduler: compile started or finished, new production day, master paused or resumed, primary master started, backup master took over, master shut down, lost connection to an agent, and an agent reaching its job limit. Create one with a name, a trigger and its Actions on the Associated Action(s) tab. Triggers about one agent can be narrowed to a single agent. Trigger History shows each firing. Triggers marked "not raised by this scheduler" can be saved but never fire.
- A new System Event starts disabled; enable it when it is ready. New events default to the Workgroup you used last.

Build a pipeline with orchestrations

Run a pipeline of steps over your existing jobs, with decisions, loops, approvals and retries.

- Orchestrations are optional. An administrator turns them on under Administration > System Settings > Enable Orchestrations (this also needs a licence that includes them). Until then the menu items below are not available.
- Open Definitions > Orchestrations and create one. Drag steps from the toolbox onto the graph, or edit the same pipeline as YAML; both produce one definition. Every save is a new immutable version, and each run records the version it executed.
- Steps include: run a Job or Job Group (with optional parameter, agent and runtime-user overrides), an inline script (cmd, PowerShell, Python, sh or bash) on an agent, a built-in tool (nearly 30 ready-made steps for files, archives, checksums, ping, port and HTTP checks, downloads and services), a condition, a wait, an approval, a for-each loop, and a sub-orchestration. The command library panel lists the tools.
- Steps pass data on: typed run parameters, and step outputs that later steps read in parameters, scripts and conditions. Choose when a step fires (all upstream succeeded, all finished, one succeeded, one failed, or all failed), retry it with backoff, and add failure handlers that run only when the pipeline fails.
- A for-each loop runs a step, or a small chain, once per item of a list, in parallel up to the limit you set. Lists can come from files in a folder, services, or JSON, CSV and XML data.
- Run an orchestration on demand or on a schedule, each with its own time zone. In Schedules... choose how to schedule it: like a Job (a Calendar says which days, a From and optional To time set the window, and reruns start it again every N minutes, N minutes after the previous run finishes, at custom times, or at minutes past the hour, with an optional limit per day and an active date range), or with a cron expression. The next five start times are shown as you edit. A file event can also start one.
- Operations > Orchestration Runs shows the graph live, with per-item loop status and each step's output (stdout and stderr). Pick the day with the date box; a run still in progress stays on today's list even if it began before midnight. Approve, retry or cancel from there.
- An approval step waits for a named approver or workgroup, in the console or by email, where each approver gets their own signed link.
- Choosing a file or folder in a tool step has a Browse button for the target agent's disk.

Build, test and deploy (CI/CD)

Turn a git push into a tested, approved deployment - with the result shown on the commit.

- An orchestration can run a whole build pipeline: a push to GitHub or GitLab starts a run, the code is checked out into a working folder on a build agent, built and tested, packaged, and deployed to a protected environment after approval - and the result shows on the commit. The quickest start is From template... > Build & deploy next to New: pick an agent on each step, adjust the build and test commands and save.
- Start runs from git with a webhook (Administration > Webhooks, Starts: An orchestration, called by GitHub, GitLab or any system with a bearer token). Branches and Events limit which pushes start a run. Each run knows the push's branch, commit, author and message as `${trigger.branch}`, `${trigger.commit}`

and so on, and the orchestration's parameters can be filled from them. Each run also has its own number (#1, #2...), `${run.number}`, handy as a build number.

- Run settings... > Workspace gives each run its own working folder on every agent it uses (<orchestration> <run id>). Script and tool steps without a working directory start in it, so checkout, build and test steps share the same files. Its path is `${run.workspace}` and `IDRUNS_WORKSPACE`. When the run ends the folder is removed: always, only after success, or never.
- The Git checkout tool fetches one branch, tag or commit (for example `${trigger.commit}`) into the workspace. For a private repository put the token in the step's Environment as a secret and give that variable's name; git receives it through its environment, never on its command line or in the output. The agent needs Git installed.
- Keep tokens, passwords and keys as secrets (Secrets... in the orchestrations list). A secret belongs to a Workgroup or, set by an administrator, is global; a Workgroup's own secret wins over a global one of the same name. A value can be replaced but is never shown again.
- A script or tool step gets environment variables under Environment: plain text (parameters and step outputs allowed) or a secret. The script reads them like any variable (`$env:NAME` in PowerShell, `$NAME` in sh). A secret is never written into the script or the run, and shows as `***` in the step's output. Every step also gets `IDRUNS_RUN_ID`, `IDRUNS_RUN_NUMBER`, `IDRUNS_ORCHESTRATION`, `IDRUNS_STEP`, `IDRUNS_TRIGGER` and, from a webhook, `IDRUNS_BRANCH` and `IDRUNS_COMMIT`.
- A running step's output fills in live on the run page (agents send it every 2 seconds for orchestration steps, every 10 for ordinary jobs; agents need the latest version for the faster rate). Follow keeps the newest lines in view - scroll up to read and it pauses, scroll back to the bottom and it resumes.
- Publish test results (a tool) reads the JUnit XML files a test run wrote - JUnit / Maven, `pytest --junitxml`, `jest-junit`, `.NET JunitXml`, `go-junit-report` - and shows the totals and each failure, with its message and stack trace, on the run page; the files are kept as an artifact. Give it needs on the test step with trigger rule `all_done` so failed tests are reported too. It fails when a test failed (unless that option is off), and later steps see `${steps.output.failed}`, `.passed` and `.total`.
- Publish artifact zips a file, folder or wildcard from a step and keeps it on the server under a name; Fetch artifact downloads it on any agent later in the same run and unzips it, so a build on one server can be deployed from another. The run page lists each artifact for download. Artifacts are kept for 30 days (`ORCHESTRATION_ARTIFACT_DAYS`), up to 2 GB each.
- Environments... (in the orchestrations list) defines deploy targets such as staging and production. A step whose Deploys to environment is set runs only for the environment's allowed branches (the webhook's branch or tag, else the orchestration's branch parameter), waits for one of its approvers when it has some (by email link, or Approve on the run page - only they or an administrator may decide), and gets the environment's own secrets in place of same-named Workgroup or global ones, so `${secrets.db_password}` can differ per environment (`IDRUNS_ENVIRONMENT` is set). Every deployment is kept in the environment's history, which shows what is deployed there now.
- Run settings... > Concurrency keeps runs of the same group from overlapping - e.g. group `deploy-${inputs.environment}` allows one deploy per environment at a time. The group may use `${inputs.}`, `${trigger.}` and `${run.number}` (empty = the orchestration's name) and is shared by every orchestration using the same text. A newer run either queues (shown as QUEUED behind the older run), cancels the older runs first (the newest push wins), or is skipped (recorded as Cancelled).

- A GitHub or GitLab webhook can report each run back on the commit, and its pull or merge request, as a check: pending, then success, failure or cancelled, linking to the run. In the webhook choose Report status and an orchestration secret holding the token - GitHub: a token with Commit statuses read and write; GitLab: an access token with the api scope. Check name defaults to IDRuns/. A failed report is shown on the run page and retried, and never affects the run.
- Server setup: artifacts, test results and the links in commit statuses need PUBLIC_BASE_URL in the server's .env - the address agents and git hosts reach the server at, e.g. <http://scheduler:8010>. GitHub or GitLab must be able to reach the webhook's URL.

Move files between agents (File Mover)

Deliver files of up to 10 GiB between agents with verification and recovery.

- File Mover is a Plugin Job type (Plugin Job > File Mover). It copies or moves a file through the server to one or more agents on Windows or Linux.
- Choose the source from a file event or an explicit agent and file. Each destination needs an agent and a full target filename. A collision policy decides what happens if the file exists: Fail, Skip identical, or Replace with backup.
- A delivery stages, verifies, backs up and then publishes the file. Deployment templates also stop and start services around the publish and require an HTTP health check afterwards. Every transfer is checked with a SHA-256 hash, and an interrupted delivery recovers on its own.
- Administration > Connections shows each agent's File Mover status: Ready, Needs token, Offline or Not supported.
- By default an agent and Runtime User only need to be attached to the job's workgroup. An administrator can turn on Administration > System Settings > Require File Mover path grants; transfers may then only use folders granted to the workgroup (Workgroups > File Mover), and a Runtime User needs a grant that names it. This applies to File Mover jobs and to file event and orchestration copy or move-to-agent steps.
- A destination can be an agent on another IDRuns server (for example a DR site). Add the server under Administration > Connections > Plugins > File Mover (its address plus an API token, or a username and password), then on the job's Destinations tab set Where to Agent on another IDRuns server, choose the connection and give the agent's name there. The other server stages, verifies, backs up and publishes on its own agent under its own workgroup and path-grant rules; the file is held in both servers' relays until the workflow ends.

Operations

Use live operational views to monitor the scheduler and investigate problems.

- Job Activity shows the runs for a production date; Event Activity shows event processing; Orchestration Runs shows pipeline runs for a day.
- Alerts lists conditions requiring attention. Select alerts and choose Respond to Acknowledge them (they stay open for tracking) or Close them with a response.

- Alerts also report SLA deadlines that are at risk or missed. When AI is enabled and configured, a failure alert includes a short AI diagnosis of the likely cause.
- Logs provides diagnostic history. Search by job, run, message, or time. In a run's log messages you can limit the trail to a single day, which helps for a job carried over several days.
- Schedules summarizes each production date: total, scheduled, ad hoc, carried-over, failed, and succeeded jobs. Double-click a date to open its Job Activity.
- Master Status shows the scheduler's current state, what each connection is running or has queued, and recent scheduler activity.
- The top bar shows the number of waiting and active jobs and the success rate of the runs that have finished today (hover it for the counts), and refreshes automatically.
- To enable, disable, insert into the schedule, remove from the schedule, or move several jobs at once, check them in Definitions > Jobs.

Work with tables

Shared grid controls behave consistently throughout IDRuns.

- Click a column heading to sort. An arrow indicates the current direction.
- Use Search to match values across the visible data, or use column filters for precise filtering.
- Drag column edges to resize and column headings to reorder when those controls are enabled.
- Checkboxes select rows; Shift-click selects a range. The select-all checkbox applies to the currently filtered rows.
- The icons above a table apply to it: add a record, delete the selected records, recompile, and expand or collapse groups. A sort-override icon appears when a column sort overrides the dependency order; select it to clear the sort.
- Show-checked displays only the checked rows; hide-checked removes them. Turn either off again to see everything.
- Double-click opens the primary record action on screens that support editing or drill-down.
- Sorting, filters, column sizes and order, and the show/hide-checked choices are remembered per table in this browser. Checked rows are remembered on Connections, Jobs, and Job Activity; a checked item that no longer exists is dropped automatically.

Create and maintain definitions

Definitions describe what runs, when it runs, and what it depends on.

- Jobs contains job groups and jobs. Groups organize inheritance for agents, agent lists, calendars, time windows, variables, owners, and runtime settings.
- If you cannot change a job, double-clicking it still opens its definition, marked (read only) in the title.
- Select the dependency-lines icon above the Jobs grid to draw lines between jobs and the jobs they depend on. Dashed grey lines are dependencies in another Job Group. Inside Go Into Group, dependencies on jobs outside the group end in a small circle; hover it to see the outside job.

- Create OS command, FTP, Terminator, Approval Gate, Branch, and Plugin jobs. Plugin jobs can run in the server process or on an assigned agent when the selected plugin supports it.
- SCP jobs copy files to or from another server over SSH: pick a connection (server and key or password), a direction, the paths and the scp options from a form, and the job runs `scp` on its agent. The form shows the exact command. See `docs/SCP.md`.
- Calendars define eligible run dates; Fiscal Calendars add reusable business periods. Actions and Events define reusable responses and triggers.
- Global, group, and local Variables can be inserted into supported fields. Review inherited values before overriding them locally.
- Workflow Designer visualizes dependencies. Recycle Bin contains deleted definitions that may be restored. A deleted Job Group that still has runs scheduled shows Still scheduled there and cannot be permanently deleted until its future runs are removed or done. Restoring a Job Group asks from which date it goes back on the schedule; Cancel leaves it restored but off the schedule until the next compile.
- Save changes only after validating commands, credentials, calendars, dependencies, and expected ownership.

Schedule and run work

Compile definitions into runs and control one-off or grouped execution.

- A job becomes eligible from its Calendar, effective-date range, timezone, time window, and dependencies. Parent-group settings are inherited unless overridden.
- Use Insert in Schedule on one job, a whole group, or a multi-selection to create controlled runs with optional date, time, agent, and command overrides.
- Use Remove from Schedule only for runs that should no longer execute. Both actions are on the Definitions > Jobs toolbar and right-click menu. Jobs must be enabled to be scheduled; enabling or disabling a Job Group offers to apply the change to everything nested inside it. Disabling or deleting a Job Group asks from which date to take it off the schedule: its not-yet-started runs from that date on are removed, earlier days stay scheduled (a deleted group keeps showing in Job Activity on those days), and Cancel leaves its runs scheduled as they are. Whatever is nested inside it follows the same date.
- Recompile in Job Activity rebuilds a production date from the current definitions and keeps runs you inserted manually. It is administrator-only.
- Use Operations > Schedules to review each production date's totals and Job Activity to follow each compiled or manually inserted run.

Plugins and connections

Administration. Configure reusable external-system connections and build plugin-backed jobs.

- Open Administration > Connections. Agents lists scheduler workers; Plugins contains named connections supplied by installed plugins; under Plugins, the JDBC Drivers sub-tab holds the driver jars the JDBC plugin uses. Non-administrators can view these screens read-only.

- Add an agent, or double-click one to edit it. Its Communication tab shows recent work sent to it and results received.
- To delete agents, check them and delete, or open one and choose Delete Connection. The built-in Master connection cannot be deleted, and an agent with launched or running jobs cannot be deleted until those finish or are cancelled.
- Create a plugin connection, enter its configuration and credentials or Runtime User, and use Test connection before saving. A connection that fails its test can still be saved or deleted.
- To delete a plugin connection, select and delete it, or open it and choose Delete Connection. A connection still used by a job cannot be deleted; the message lists the jobs to update first.
- REST API connections can store a base URL and an optional OpenAPI document. SQL Server connections can use direct server execution or agent execution when configured.
- **JDBC (any database).** The JDBC plugin runs SQL against any database that has a JDBC driver. The IDRuns server needs a JDK (11 or later); set JAVA_HOME (or JDBC_JAVA_HOME) in the backend environment if Java is not on the PATH. An administrator first opens Connections > Plugins > JDBC Drivers, chooses Add, optionally picks a common database to fill in the driver class and URL template, and adds the vendor's driver jar file(s). Then create a plugin connection of type JDBC, choose the driver, and enter host, port and database (or a complete JDBC URL), any extra driver properties, and credentials or a Runtime User (its Plugin password). Use Test connection to confirm. A JDBC job runs a query or stored procedure; turn off Commit changes to run in a transaction that is rolled back. JDBC jobs run from the server, not from an agent. A driver used by a connection cannot be deleted. Driver jars run as code on the server, so only administrators can upload them.
- Create a Plugin Job from Definitions > Jobs, choose its plugin, execution mode, saved connection, and operation-specific parameters.
- The built-in Master connection shows the backend server hostname, operating system, and detected IANA timezone. Its capacity is the system-wide active-job cap.
- The Connections screen remembers the tab you were on, the connection you last selected, and any rows you had checked.

Design a workflow

Build and review job relationships visually.

- Open Workflow Designer and choose the job group you want to inspect.
- Use the canvas to understand upstream and downstream dependencies before making changes. Align Horizontal, Align Vertical, and Fit to Page tidy the view; Save Layout keeps your arrangement.
- Use the Toolbox panel on the left: drag a Job Group, OS Job, FTP Job, Terminator Job or Plugin Job onto the canvas to create it at the drop point, then open a node to edit it. The + buttons on the toolbar still work.
- The Definitions tab of the same panel shows the Job Group tree. Drag an item onto a group to move it there, or drop a new job type on a group to create it inside. Double-click a name (or press F2) to rename it in place. Items already on the canvas are shown greyed out.

- Select Critical Path to outline the longest dependency chain in the group in red; hover the button to see its estimated length in minutes. Select Hide Critical Path to turn it off.
- After editing, verify the resulting schedule and monitor the next run in Job Activity.

Reports and historical analysis

Review execution volume, event and log activity, and status trends.

- Job Executions by Hour shows when job workload is concentrated. Internal checks (event and file triggers) are not counted.
- Expected Starts by Hour shows, for a date and by hour, the starts that already happened (the same count as Job Executions by Hour) and the starts still to come, split into runs that will run and runs at risk (the Rest-of-Day Forecast projection). Internal checks, such as event and file triggers, are not counted as job starts.
- Events by Hour and Log Messages by Hour highlight activity patterns and unusual spikes.
- Job Counts by Status compares completion outcomes across production dates.
- Executions by Job Group breaks executions down by group.
- Rest-of-Day Forecast projects what is still to run today: each waiting job's expected start and end (from its median run time over the last 14 days and the jobs it waits on), how many starts to expect each hour, and an Issues ahead list - jobs that won't run today (on hold, waiting on a job that failed or isn't scheduled, an offline or stopped agent, a time window that closes first, the day ending first) and jobs at risk (running long, past their timeout, waiting for an approval or a file/variable). It refreshes every minute.
- SLA Status lists every run of a production date whose Job or Job Group has an SLA, by where it stands against its deadline: Breached, Projected to miss (the day forecast says it will be late or won't run), At risk (80% of the window gone, or a failed run that needs a rerun), On track, Met, or No deadline yet. Click a status card to show only those runs, pick another production date, and click a run to open it in Job Activity.
- Use operational grids for individual-run investigation and reports for trends across a wider period.

Administration

Administration. Configure users, agents, connectivity, availability, and system-level behavior.

- Connections define agents and plugin-backed external systems. Runtime Users provide execution identities; Interactive Users control who can sign in and their role.
- Administrators can rotate a Runtime User password without opening the full form: select the user in Administration > Runtime Users and choose Change password... (toolbar button or right-click), pick the job type, and enter the new password twice. Only that job type changes, and the change is recorded in the audit log without the value. Scripts can use the REST endpoints POST /api/runtime-users/{name}/password and POST /api/runtime-users/rotate-passwords (several at once, all-or-nothing); both are administrator-only.

- API Tokens create bearer tokens so scripts and integrations can call the REST API without a browser login. A token is shown only once, when it is created.
- Webhooks let an outside system start one Job or Job Group with an HTTP POST. Create a webhook, choose what it Triggers, and copy the token and example curl command; like API tokens, the token is shown only once. It can trigger only that one target, and calls are rate limited and audit logged. Regenerate Token replaces it and stops the old one immediately. Turn on Allow variables to let a call send a JSON body such as {"variables": {"FILE": "orders.csv"}}; it sets the target's own Local (Job) or Group (Job Group) variables for that run only, without changing the definition. The form lists the variables a call can set. Unknown names, values over 4000 characters, and values containing <, > or line breaks are rejected, and the values passed are recorded in the audit log. A webhook can instead start an orchestration, for example from a git push. Choose Starts: An orchestration, then who calls it: GitHub (the call is signed with a secret), GitLab (secret token) or any system with a bearer token. Copy the URL and secret into the repository's webhook settings; the secret is shown only once. Branches (such as main, release/*) and Events (push, tag, pull_request) limit which pushes start a run; any other call answers 202 without a run. Each run knows the push's branch, commit, author and message as \${trigger.branch}, \${trigger.commit} and so on, and the orchestration's parameters can be filled from them or from a field of the JSON body.
- Resource Pools define shared limits (Max concurrent units) that jobs and groups can require. See Control runs, resources, and deadlines.
- AI Monitor stores the AI API key and model used by Describe Your Job..., Plan with AI, and AI diagnosis on failure alerts, and reviews job health on a schedule; the operators listed there are emailed when a new Critical finding appears. The Model list is loaded from Anthropic for your key, so it shows the models you can actually use; choose Other to type a model ID. Use Test Connection to check the key and model. It appears after an administrator turns on Enable AI in System Settings.
- Workgroups own definitions and control who may edit them; LDAP Settings configures directory sign-in. Agent Lists control which agents can execute assigned work.
- Bulk Field Update changes one field (Agent, Agent List, Calendar, Time Window, Owner, Command, or Runtime User) on many jobs at once. Match by exact or contains text, and review the result.
- High Availability configures the two scheduler servers (name and URL each). The first to start is active, the other waits on standby; **Make ... active** switches them. Confirm connectivity and role ownership before changing production settings.
- Export / Import moves IDRuns definitions and related objects between systems. Import from TIDAL maps existing scheduler content; preview and test a small scope first.
- **Moving definitions between IDRuns systems.** Open Administration > Export / Import. The top section, "Transport from another IDRuns system", works over the other system's REST API: enter its URL and sign in with an Admin username and password (not remembered; only the username is) or an Admin API token, then choose a direction. "Send selected definitions TO the remote system" (the default) lets you tick Job Groups and Jobs on this system, then "Review and send": the remote system matches the names (its own mapping sets, agents, runtime users and workgroups are the ones offered) and imports them. "Fetch selected definitions FROM the remote system" does the reverse, with the names matched on this system. You always pick the Job Groups and Jobs to move; there is no "everything" option. Tick "Include related" for the Calendars, Job Events and Actions they use, or "Include the entire remote catalog" for everything: Calendars, Job Events, Actions, Agents, Agent Lists, Variables, and the File, Variable and System Events. For systems that cannot reach each other, the "Files" section below does the same through a JSON file you

download on the source (again with a selection) and upload on the destination. Tick "Also import Calendars, Job Events, Actions..." to bring the related objects along. Imported File, Variable and System Events arrive disabled so nothing fires before you have reviewed them.

- **Names that differ between systems.** Agents, agent lists, runtime users and workgroups are matched by name. After a file is loaded, the "Names on the source system" list shows where each name stands: matched by a mapping set, the same name here, created by this import, or needing a choice. Pick a destination for each one that needs it (or leave it blank / without an owner), then Start Import; the import stays blocked until every name is decided. Under "Save these mappings as" type a unique name (e.g. "DEV to PROD" - one per source and destination pair) and click Save mapping set; pick that name under Mapping set to reuse it later. Saving under a new name copies the selected set's mappings plus your new choices; saving under the selected name updates it. A mapping set is a named list such as "DEV to PROD"; Manage... creates, edits, deletes, downloads (a small .idmap.json file you can send to another system) and uploads them. The same set can also be applied when exporting ("Rename for the destination with"). Mapping sets hold names only - runtime user passwords are never transferred, so the runtime user must already exist on the destination. A calendar, action or event whose workgroup is left blank is skipped, because those must be owned by a workgroup. Existing mappings saved for "Connect to NewWave Scheduler" were moved into a set called "Remote import (migrated)".
- Export to TIDAL pushes one or more top-level Job Groups, each with everything nested in it, to a Tidal Enterprise Scheduler as new objects. Tick the groups you want; exporting a group takes its whole subtree. Only OS command jobs are exported, and it writes to a live system, so try it against a non-production server first.
- Import from TIDAL, Export to TIDAL, Schedule Simulator, Job Iteration, Version History, Orchestrations, and AI Monitor appear only after an administrator enables them in System Settings.
- **Imported Job Groups arrive disabled.** Every Job Group Import from TIDAL creates - including the "Attach under" group, when it has to be made - is disabled whatever its Active flag was in TIDAL, so nothing it brings in is scheduled until you have reviewed it and enabled it (Jobs inside keep their own flag). A group that already exists keeps its current enabled state when re-imported.
- **Import from TIDAL mapping sets.** The Tidal agent, agent list and owner mappings you save on Import from TIDAL can be kept under a unique name - one per Tidal server and IDRuns system combination, e.g. "TES Prod to IDRuns Prod". Pick the set under Mapping set (or create one, optionally starting from the mappings shown now); mappings you save go into the selected set and Preview / Import apply it. Pick the same name again later to reuse those mappings. With no set selected the original global mappings are used. The sets are the same ones Export / Import lists, so they can also be downloaded as a file.
- System Settings controls retention of job activity and logs, the schedule compile horizon, carry-over of unfinished work, confirmation before discarding edits, and the shared Application layout. Changing the layout affects everyone, while each user keeps their own color-scheme preference.
- License shows entitlement and host binding.
- When enabled, Schedule Simulator snapshots definitions and advances virtual time without running commands, contacting agents, or changing production scheduler records.

Troubleshooting

A practical sequence for diagnosing unexpected behavior.

- Confirm the production date and selected activity date, then clear search and column filters.
- Check Alerts first, then Logs for the affected job or run ID.
- Verify the job calendar, dependencies, agent assignment, runtime user, and effective inherited values.
- For waiting work, read the specific waiting status and the Reason list on the run, then inspect prerequisite status, time windows, and Resource Pool usage. For failures, read the run message and agent logs.
- Record the job number, run ID, timestamp, status, and exact message before escalating the issue.