reportworq

Administrator Guide

Enterprise reporting, distribution, and governed AI - product documentation

Version 6.0.0.0

Contents

Administrator Guide

System requirements

reference administrator

This page lists what a server needs to run Reportworq: the operating system and hardware, the browser used for the admin console, and the network and firewall requirements. Reportworq runs on Windows Server and on Linux. Confirm these before you install, since the most common "jobs cannot read or write" and "cannot reach the server" problems trace back to the network and service-account setup rather than the software itself.

To decide which deployment shape fits your environment (Windows, Linux, or a container), see Deployment topology.

Operating system

Platform	Supported versions	How it installs
Windows Server	2016 or later. Windows Server 2012 R2 and earlier are not supported.	The MSI installer plus the Reportworq 6 Bootstrap Service.
Linux	A current 64-bit distribution, such as Ubuntu.	A tarball, or a Docker / Azure App Service container image. There is no installer and no Windows service.

Hardware

Requirement	Value	Notes
CPU	4-core, 2 GHz or faster, x64	
RAM	32 GB minimum	Raised from earlier releases for the expanded functionality in this version. Allow more for large Excel files, and about 2 GB per concurrent job on top of the base, see Server configuration.
Disk	10 GB free	Grows over time with the Repository, run history, and logs.
Display	1280x720 minimum, 1920x1080 recommended	For the administration console.
Browser window	1280x720 minimum	The size of the *browser window*, not the monitor. See Browser window size below, because the two are often different.
Browser	A Chromium browser (Chrome or Edge)	The web UI is built for Chromium.

Browser window size

The Reportworq web interface is designed against the size of the **browser window**, not the size of the monitor. Those are usually different, and the difference is larger than most people expect, so size the window rather than assuming a large monitor is enough.

Minimum 1280x720. Recommended 1920x1080. Below 1280 wide, some screens become cramped and wide tables are harder to read.

Three ordinary things make a browser window smaller than the monitor it is on:

- **Display scaling.** Windows scaling shrinks the usable area the browser reports. A 1920x1080 monitor at 150% scaling gives the browser 1280 pixels of width; at 125% it gives 1536. Enterprise laptops are frequently scaled by default, so a 1080p screen often behaves like a 1280-wide one.
- **Browser toolbars.** Tabs, the address bar and the bookmarks bar take roughly 90 to 140 pixels of height. A 1366x768 laptop screen leaves about 630 pixels of usable height.
- **Windows that are not maximized.** Running Reportworq beside Excel or Teams on a wide monitor halves the width available to it.

Reportworq's own navigation rail takes a further 280 pixels of width, which does not change with the window size. Collapsing the rail with the menu button at the top left returns about 170 pixels, which is worth doing on a narrow window or when working with a wide table such as Run History.

If the window is very small, Reportworq shows a banner telling you so. Widening the window, maximizing it, or collapsing the navigation rail resolves it.

Additional requirements on Linux and in containers

The Linux tarball and the container image bundle the .NET runtime, so no runtime prerequisite is installed on the host. CPU, RAM, and disk are the same as on Windows. Two host-level items are specific to Linux.

Requirement	Value	Notes
OS libraries	`fontconfig`, `libfontconfig1`, `libfreetype6`, `libgomp1`	The native dependencies of the rendering engine.
Fonts	At least one font package, for example `fonts-liberation` or `fonts-dejavu`, or your corporate fonts	Not optional. A clean Linux server has almost no fonts, and report output renders blank or boxed without them. The container image ships with fonts already installed.

Linux is at feature parity with Windows. There is no capability a customer uses that the Linux build lacks. The only piece not built for Linux is the Windows service host, which is a launcher and is replaced by the systemd unit or the container entry point.

Running version 5 and version 6 on the same server

Reportworq 5 and Reportworq 6 can run **side by side on the same machine**. Version 6 installs into its own folder (`C:\Program Files\Reportworq 6` by default) and listens on its own default port, so it does not collide with an existing version 5 installation, which keeps its own folder and its own default port.

This is what makes a staged migration practical: you can stand version 6 up next to a running version 5, move content across, and validate before decommissioning anything. When you are ready to bring version 5 content over, use the import path rather than pointing version 6 at a version 5 repository, see First-run setup.

Network and firewall

Reportworq hosts its web application and job runtime on a single service. Two TCP ports must be open in the firewall on every server that runs Reportworq, including each load-balancer node in a cluster. The port numbers below are the Windows defaults; on Linux

and in containers the default web port is **8080** and the Event Hub follows on **8081**.

Port	Purpose
8600	The main web server. This is the port administrators and users connect to.
8601	The Real-time Event Hub bus. Reportworq uses the main web port plus one.

The default web port is **8600** (with the Event Hub on **8601**). Earlier releases defaulted to 8300, and Reportworq 6 uses 8600 so it can run alongside an existing Reportworq 5 on the same server without a port conflict. If you are upgrading from v5, are not running v5 and v6 side by side, and your firewall is already open for 8300, you can change the web port back to 8300 in Server configuration; the Event Hub port then follows as 8301.

Warning: opening only the web port is a common mistake. Local access on the server works immediately, but a cluster that opens only the web port will connect while its real-time coordination fails silently. Open both 8600 and 8601 on every server. If you change the web port later in Server configuration, the Event Hub port moves with it (web port plus one).

Local access on the server works as soon as the service is running. Remote access is blocked until you add an inbound-allow firewall rule for the web port. After adding the rule, test remotely with the server's name and port, for example `https://<computername>:8600`.

The server also needs outbound internet access during installation, configuration, and use, for license activation, updates, hosted AI, and CloudHub. For an air-gapped server, see the offline activation path in Install Reportworq.

HTTPS out of the box

On a fresh Windows on-premises install, Reportworq turns on HTTPS automatically at first boot. It generates a self-signed certificate, trusts it on the server, and enables SSL before you sign in, so the first browse to the server is `https://localhost:8600` and local access on the server works right away because the certificate is already trusted there.

- Remote browsers do not trust a self-signed certificate until you distribute the trusted-root certificate to them.
- For production, replace the self-signed certificate with one from a trusted certificate authority.

For the certificate actions, see Server configuration.

This automatic step is Windows-only and fires only on a genuinely new install. Linux and container hosts do not self-provision a certificate; enable HTTPS there through

configuration or a TLS-terminating reverse proxy, see Deployment topology.

HTTPS is required for the Excel add-in and Microsoft 365

HTTPS is mandatory for the Excel add-in and for every Microsoft 365 feature (Entra sign-in, SharePoint, Microsoft 365 email, and Teams distribution). Over plain HTTP, add-in sign-in and the Graph-based integrations will not work, and they fail quietly rather than reporting an error. Because a fresh Windows install already serves over HTTPS (see above), these features work from the first boot on Windows; on Linux and container hosts, make sure HTTPS is configured before you rely on the add-in or Microsoft 365 features.

Notes and limits

- The MSI installer and the Windows service are Windows-bound. The Linux tarball and the Docker / Azure App Service container image run the same application with no installer and no Windows service, see Deployment topology.
- The RAM figure is a stated requirement, not enforced by the software at startup. As a rule of thumb, allow about 2 GB per concurrent job on top of the base. The default is 4 concurrent jobs, and the concurrency setting is in Server configuration.
- **No graphics requirement.** Earlier Reportworq documentation listed DirectX 12. That requirement belonged to the older rendering path and does not apply. Reportworq needs no GPU and no display adapter on any platform. What that row was standing in for is fonts, which are a real requirement, see the Fonts row above.

> **Going deeper.** For multi-node clusters, cloud hosting, and how the two ports apply across a load-balanced set of servers, see Deployment topology. To turn these baseline figures into a server count for a real workload, see Distribute the load.

Accounts, groups, and entitlements

how-to administrator

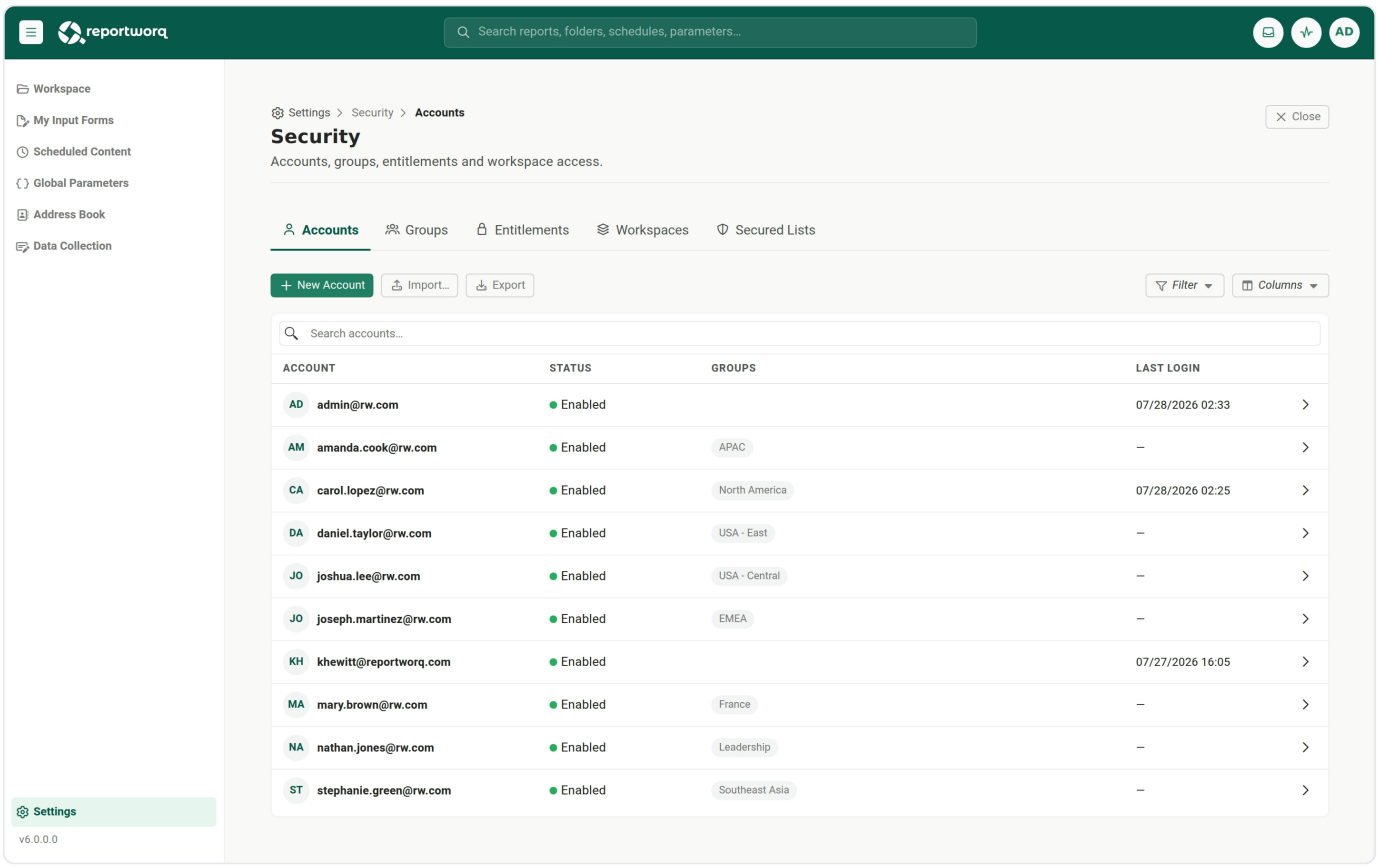
Settings ▸ Security is where you define who can sign in to Reportworq and what they can do. It has a five-tab strip: **Accounts**, **Groups**, **Entitlements**, **Workspaces**, and **Secured Lists**. This guide covers the first three, plus the accounts import. Workspaces and their membership are covered in Workspaces, and row-level security in Secured lists.

An account authenticates. A group collects accounts so you can grant access once to a whole audience. An entitlement is a role that grants a capability, and most roles consume a license seat, so plan them deliberately.

When to use it. During onboarding and offboarding, when forming reporting audiences (groups), and when delegating administration.

Before you start

- You need Administrator rights. The entire Security surface is administrator-only.
- The active authentication provider (Native by default) supplies the account store. See Sign-in and SSO.
- Know your licensed seat counts before you assign roles. See Licensing and entitlements.



Create an account

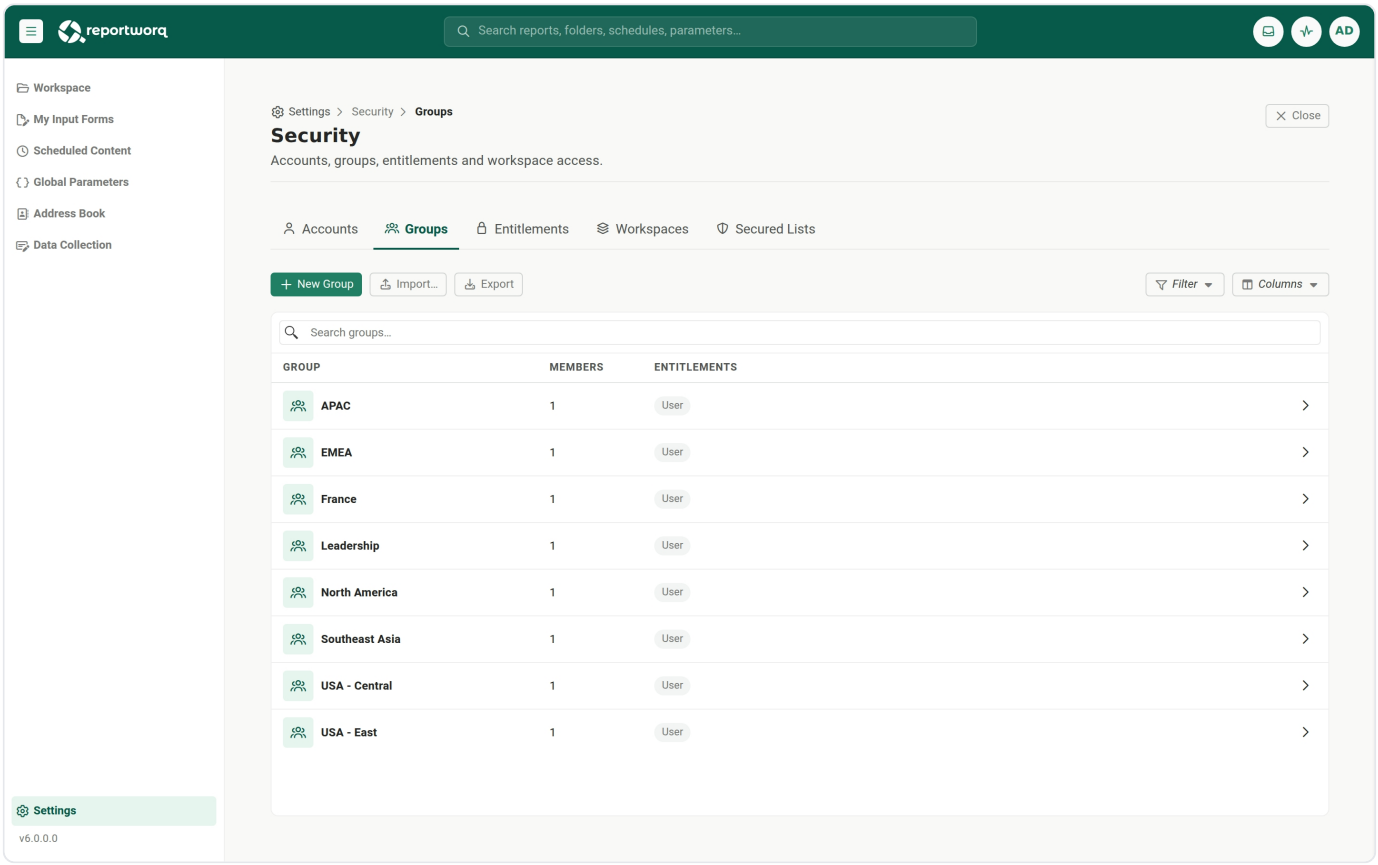
1. Open **Settings**, select the **Security** card, and stay on the **Accounts** tab.
2. Select **New Account** and enter the person's email address. The account is created as a new, enabled row.
3. To disable an account, drill into it and turn off the **Account Enabled** switch. The switch is the commit, there is no separate Save, and the list Status badge updates immediately. A disabled account cannot sign in.
4. To remove an account, drill into it and select **Delete**, then confirm the destructive prompt.

> **The account name defaults to the email.** A new account's **Name** is set to the email address you entered. To give it a friendly display name, drill into the account and edit the **Name** field. Until you do, the name stays as the email.

> **Two ways to set up an account.** You can build up an account across the Security tabs, adding it on **Accounts**, placing it in groups on **Groups**, and granting roles on **Entitlements**. Or you can drill into an individual account and edit it directly. Opening the account is where you set its **Name**, so use the direct path when you want to name the account rather than leave it as the email.

Offboard by disabling first. Rather than deleting a departing user's account right away, disable it so sign-in stops while their history and content ownership stay intact. Delete it later, after review.

Create a group and add members



1. Select the **Groups** tab, then **New Group**, and enter a name.
2. Drill into the group and use its **members** picker, a filterable multi-select of accounts. Selecting an account adds it as a member.
3. Membership edits apply to the group, not to the account. If you delete a group, its membership relations disappear with it.

Groups are the recommended way to grant access: define a **Controllers** group here once, then grant that group access to a folder in Workspace security or map it in a secured list, so membership governs who sees the content.

Dynamic groups driven by OIDC claims

A group does not have to hold an explicit member list. A group can carry a **claim string**, and membership is then decided by the claims your identity provider puts on the token at sign-in. Anyone whose token carries a matching claim is a member of that group, and

inherits whatever that group has been granted.

This is the cleanest way to run access at scale, because your identity provider stays the single place group membership is managed. It also changes three things you need to plan for.

Reportworq cannot count the members of an external group. Membership lives in your provider, not in Reportworq, and it can change without Reportworq being told. Nothing here will warn you that a claims-driven group has grown past what you are licensed for, so **keep the provider-side group under the licensed count of whatever it grants**, and audit it periodically. See Licensing and entitlements.

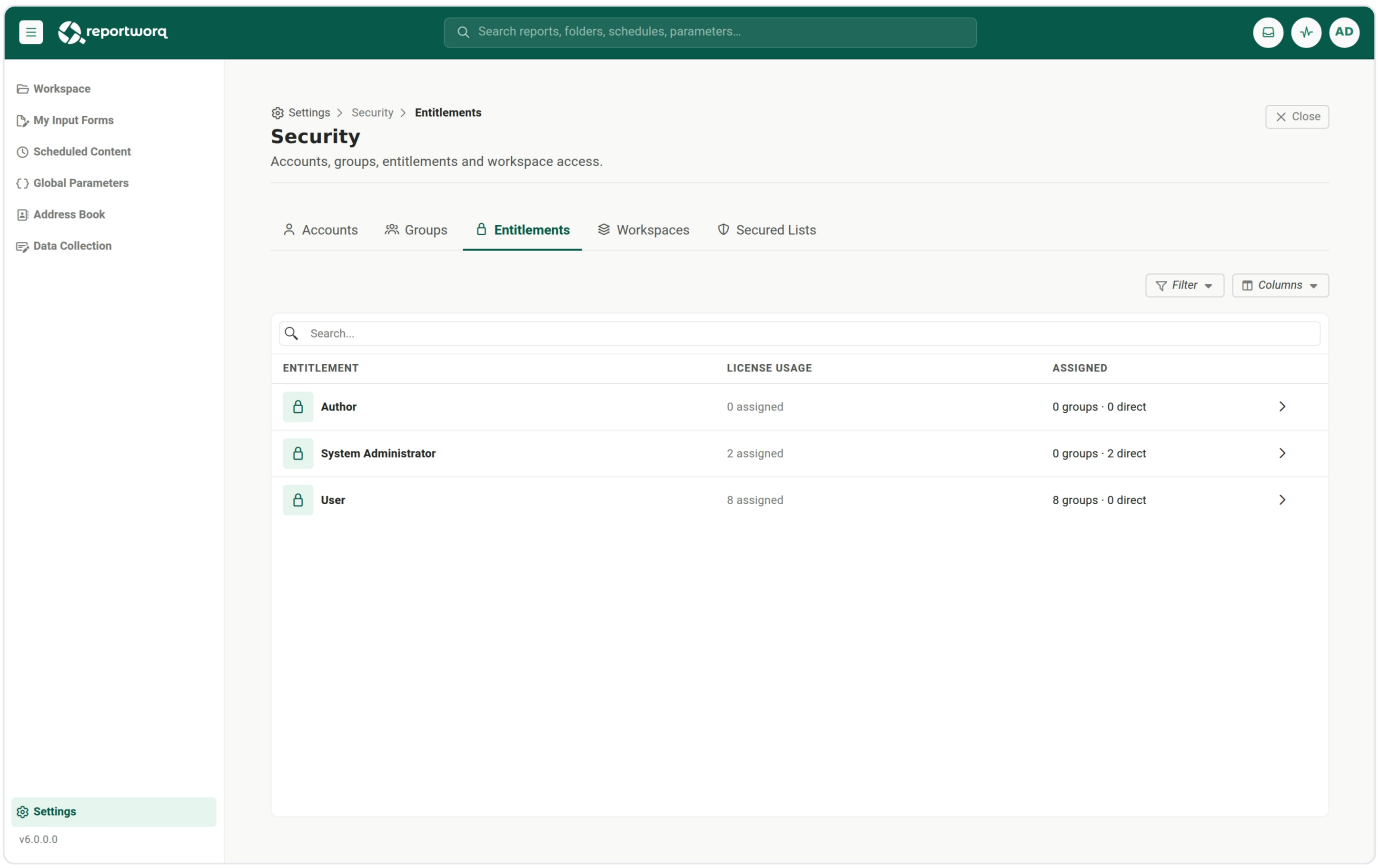
Seats are consumed at sign-in, one user at a time. A user matched by a claim is provisioned as they sign in, and the entitlement that group grants draws its seat at that moment, not when you configure the group. So an overage from a claims-driven group appears **gradually**, as people come online, and the first symptom is usually a user who cannot sign in rather than anything on the Security screen.

Changes sync at the next login, which is the trap when offboarding. If someone's provider-side group membership changes, Reportworq picks it up the next time they sign in. That is fine for a user who is still active. It is exactly wrong for a user being offboarded: **someone removed from a group in your provider may never sign in again, so their entitlement, and its seat, can sit there indefinitely.** For an offboarded user, act in Reportworq directly rather than waiting for a sync:

- **Clear the account's Last Claims**, which drops the stale entitlement immediately, or
- **Disable the account.** `Enabled` is the only gate, and a disabled account is excluded from every grant, including All Users.

Do not rely on the provider-side removal alone.

Grant an entitlement (role)



1. Select the **Entitlements** tab. A licensed instance always exposes at least one entitlement.
2. Drill into the entitlement you want to grant. Its detail carries an **assigned-groups** picker and an **assigned-accounts** picker, so you can grant a role to a whole group or to an individual account.
3. Add the group or account, and the role takes effect on that subject.

The system-level entitlements map to the three licensed roles:

Entitlement	What it grants	License seat it consumes
Administrator	Full server administration: authentication, datasources, report providers, distributors, and all of Security. Reaches every workspace automatically, and is the only role that bypasses content security.	One Administrator seat
Author	Authors Reportworq jobs (bursting and distribution), data models, and campaigns within the workspaces granted to them; gates the Excel authoring add-in.	One Author seat
User	Signs in to consume Reportworq: view output by ACL, participate in contribution campaigns as a contributor or approver (when the Contribution capability is licensed), refresh a distributed PowerPoint deck in the add-in, and reach content through the MCP server and the AI assistant.	One User seat

At least one entitlement is required to sign in. Workspace access is granted separately on the **Workspaces** tab. Everyone listed on a workspace is a member, and membership is the only level, see Workspaces.

> **The Administrator entitlement was labeled "System Administrator" in earlier versions.** It is the same entitlement under a shorter name. Exported workbooks and older license notes still use the longer form, and the export column is still `IsSystemAdministrator`.

> **Contribution and PowerPoint are capabilities of the User role, not separate seats.** There is no longer a "Contribution End User" or "PowerPoint End User" entitlement. A User participates in contribution campaigns when the installation licenses the **Contribution** capability, and any User can refresh a distributed PowerPoint deck through the add-in. See Licensing and entitlements for the installation-level capabilities.

Who can see and change content security

Content security is the list of accounts and groups on a folder, a report, or the workspace as a whole. Who may look at those lists, and who may change them, follows one rule everywhere they appear.

Role	Can see content security	Can change content security
User	No, anywhere. The **Security** tab is not offered on Properties, and the workspace security surface does not open.	No
Author	Yes, read-only, everywhere it is shown.	**No**
Administrator	Yes	Yes

An Author who opens the **Security** tab of an item, or the workspace security surface, sees exactly who has access and a notice saying the list is read-only. There is no **Save**, no picker for adding an account or group, and no way to remove one. This is deliberate: an Author builds and runs content, and knowing who can see a report is part of that job, but changing who can see it is administration.

The same rule holds however you arrive. Opening a security surface from a bookmark, a pasted link, or browser history gives you exactly the access your role allows, not the access the toolbar happened to offer.

> **"Reportworq Administrator" on a license line is not an administrator.** The **Author** entitlement is billed as *Reportworq Administrator*, and older exports and license notes use that name. It grants authoring, not administration, and it does not allow security changes. See Roles and what you can do.

All Users and the default access

Reportworq has a built-in **All Users** subject that stands for everyone who can sign in. A fresh installation grants broadly to All Users so the instance is usable immediately: the root folder's security, the default workspace, and the Administrator entitlement all start assigned to **All Users**.

Tighten these defaults as you provision real accounts and groups. Grant named accounts and teams the access they need, then remove the **All Users** grants from the root folder, from the default workspace, and from the Administrator entitlement, so access is governed by the specific accounts and groups you intend rather than by everyone.

Protect your license seats and your access

- **Each role draws from its own seat pool.** Administrator, Author, and User seats are counted separately, an Administrator seat is not an Author seat and an Author seat is not a User seat. Budget each pool against the roles you plan to assign.

- **Always keep at least one Administrator.** If you remove the last one, no one can reach Security, and you have to use the lockout recovery path in Sign-in and SSO.
- **Membership costs no seat until first login.** An entitlement granted through group or identity-provider claims is only materialized, and only consumes a seat, when the account next logs in and re-evaluates its claims. This is why a newly granted role may not show a seat used yet.
- **Release a seat immediately by clearing "Last Claims."** Because a removed role keeps holding its seat until the account next logs in, clearing the account's **Last Claims** drops the stale entitlement now, without waiting for the user's next sign-in.

Bulk-provision from a workbook

1. On the **Accounts** tab, select **Export** to download an **accounts.xlsx** workbook. Its header row is `Id · Name · Email · Enabled · IsSystemAdministrator · IsReportworqAdministrator · IsReader`, followed by one TRUE/FALSE column per group.
2. Edit the rows: set the `Is...` columns for entitlements, and the per-group columns for membership. The column names are legacy, but they map to the current roles: `IsSystemAdministrator` is the **Administrator** role, `IsReportworqAdministrator` is the **Author** entitlement, and `IsReader` is the **User** entitlement. There is no `IsContributionEndUser`, `IsPowerPointEndUser`, or `IsReportingEndUser` column in v6: those entitlements are retired, and any such column left over in an older workbook is ignored on import. Contribution participation is not set here at all, it follows from the **Contribution** capability plus the campaign's own contributor and approver assignments. Leave the `Id` blank on brand-new rows.
3. Select **Import** and choose the file. Each row is matched to an existing account by `Id` first, then by `Email`, and a new account is created when neither matches. The import is recorded in the audit log under **Import permissions**.

> **The import only adds and updates, it never deletes.** Removing a row from the workbook does not delete that account. To remove an account, delete it on the Accounts tab.

> **Never edit the `Id` column.** `Id` is the primary match key. Changing it re-targets the row to a different account or creates a duplicate. The `Id` and `Email` columns are both required, and a missing `Id` column fails the import.

Notes and limits

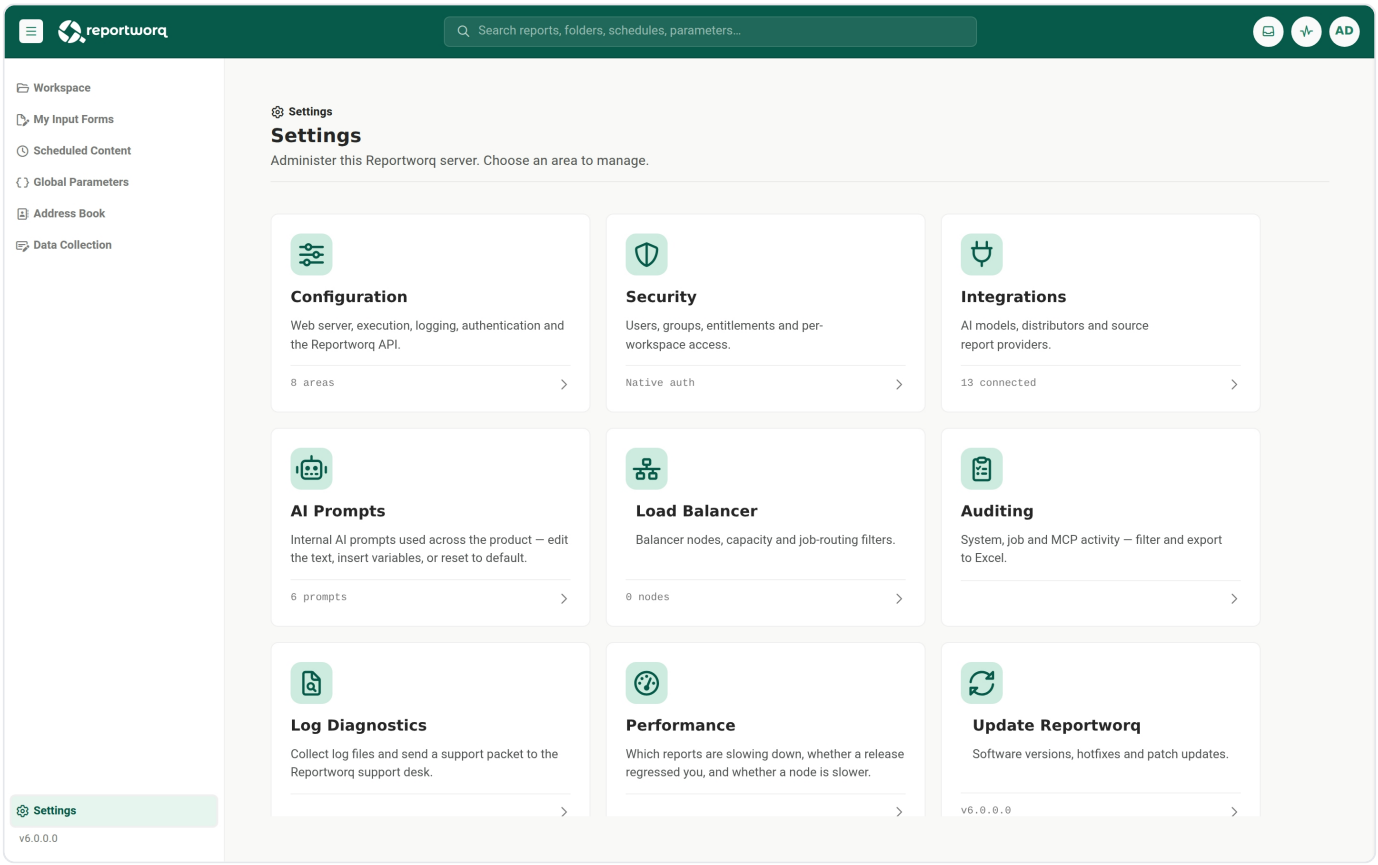
- New accounts are enabled by default.
- This surface governs identity and role, not which specific folders or reports an account can open. Per-item access is set separately: a workspace has a root security policy, and each folder and item carries its own access list on its **Properties**, under the **Security** tab. So you can secure access at the folder level, not only at the workspace level. Both are administrator-only to change, see Who can see and change content security. See also Roles and what you can do and Workspaces.
- **The workspace root policy is the top of everything.** It decides who can see content across the whole workspace, and every folder and item inherits from it unless it is overridden. Removing an entry there removes that audience's view of the entire workspace at once, so treat it as the highest-impact security change in the product. Only an Administrator can make it.
- Workspaces themselves are created, renamed, and deleted on the **Workspaces** tab of this screen, alongside their membership. See Workspaces.

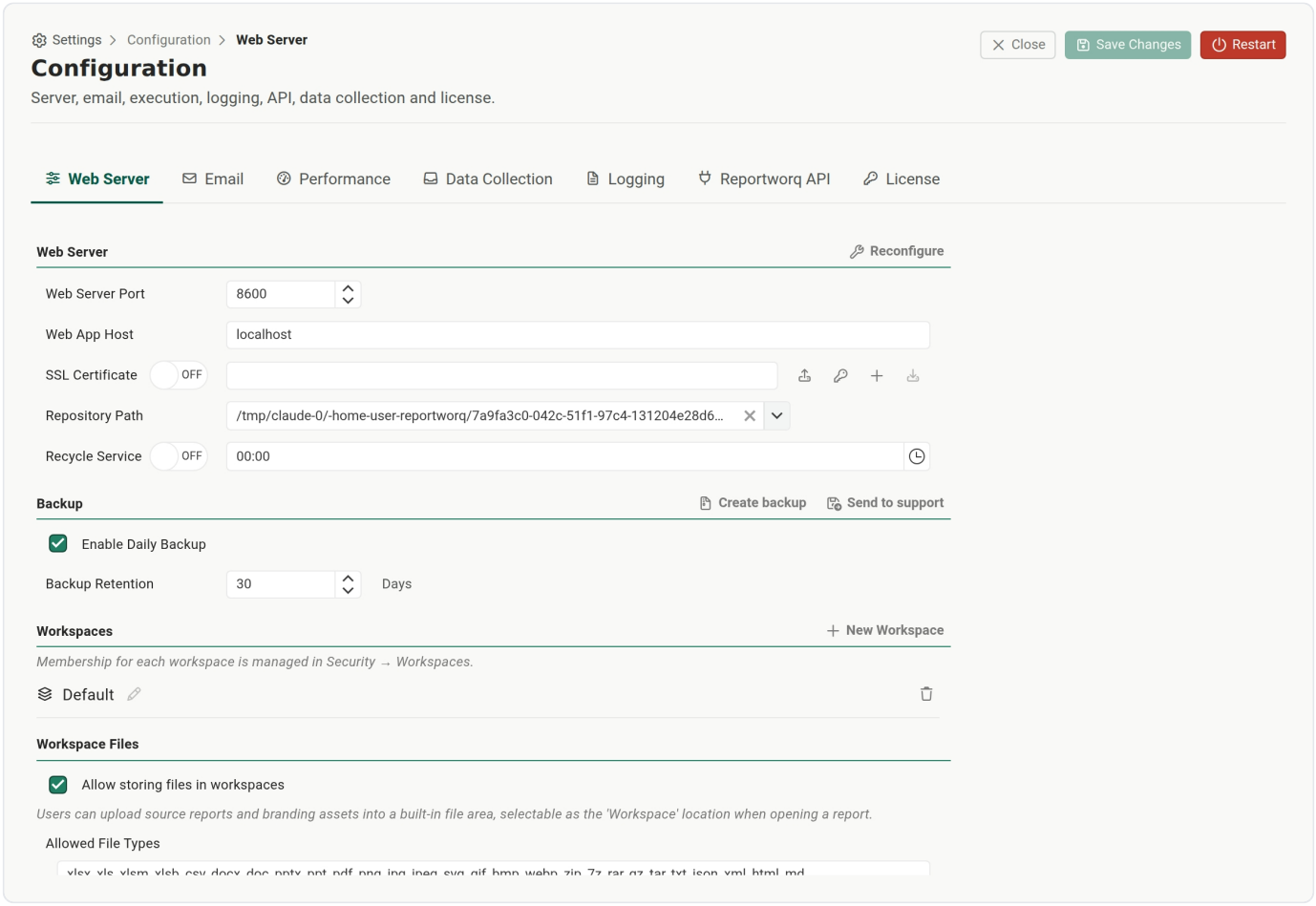
> **Going deeper.** How groups, accounts, and roles combine into what a person actually sees is the access model. See Workspaces and Secured lists.

Server configuration

reference administrator

Settings, Configuration is the administrative editor for how a Reportworq instance runs. It is a single screen with a seven-tab strip, reached from the **Configuration** card on the Settings landing. Each tab edits its own view of the settings store, and most changes take effect only after you select **Save Changes**. A few settings (the web server port, SSL, and the Repository path) also require a service **Restart**.





This page is a reference for every tab, the individual settings each one holds, and the three operational rules that cause the most trouble if you get them wrong: how peak memory scales with parallel execution, how to relocate the Repository safely, and the SSL self-signed certificate traps.

The header and the unsaved-changes guard

The screen mounts on the **Web Server** tab and shows three header actions:

- **Close** returns to the Settings landing.
- **Save Changes** stays disabled until the active tab has unsaved changes.
- **Restart** restarts the Reportworq service. This is destructive to any in-flight work, use it deliberately.

Editing a field enables **Save Changes** and marks the tab with an unsaved-changes dot. If you switch away from a tab that has unsaved edits, or close the screen, Reportworq prompts you to discard those edits. Switching tabs or closing re-reads the store, so unsaved edits are discarded rather than kept. Save what you change before you move on.

Note: administrator-level settings live here. Authentication is not on this screen, it moved to the Integrations hub. MCP and AI-agent access also live in the Integrations hub now, under **AI Agent Access**, not on the API tab. See The integrations hub.

The seven tabs

Tab	What it configures	Notes and limits
Web Server	Port, host, SSL certificate actions, the Repository path and **Reconfigure**, Recycle Service, Backup (create and send), and Workspace Files	Workspaces themselves are **not** here, they are managed on Workspaces under Settings ▸ Security. Port, SSL, and Repository changes need a Restart.
Performance	Execution (history retention, job concurrency, out-of-process execution, fonts), Contribution (log level, warm pool, session cap, recycling), Reporting (out-of-process queries, timeouts, log level), and Advanced Features (three switches plus **Clear Repository Cache**)	Tunes job concurrency, the contribution runtime, and the reporting query engine. Advanced Features is collapsed by default. For how to size these against a workload, see Distribute the load.
Logging	Logging (retention, clear/download/send logs, a live Debug Logging switch) and Audit Logging (audit retention and enablement)	Debug Logging is a live instance-wide switch, outside the save flow. Audit settings feed the audit logs.
Email	A provider dropdown (Basic, SendGrid API, Microsoft 365 SMTP, Microsoft 365 Graph), a Default From address, and a test-email action	The field set changes per provider. Graph uses the Azure app registration. The test email is blocked until email settings are saved.
Reportworq API	Local API key (created with **Regenerate**, plus Clear); for administrators, Cloud API	Enabling the Local API with no key is a save error. The Cloud API category appears for administrators only. MCP and AI-agent access are no longer here, they moved to the Integrations hub.
License	License information, status, key, and Activate / Release / Manual activation	This tab has no Save or unsaved-changes flow. Its actions take effect immediately and reach the external license service, see Licensing and entitlements.
Data Collection	Write-back settings and mailbox settings (provider, protocol, host, port, credentials)	Available only with the licensed **Contribution** capability, which includes Data Collection; otherwise an empty state notes it is not in your license.

Individual settings reference

Defaults below are the shipped defaults. Settings marked **Restart** take effect only after a service restart.

Setting	Default	What it does	Tab
Web server port	8600	The TCP port the web app listens on (Reportworq 6 default; earlier releases used 8300). The Real-time Event Hub bus uses this port plus one (8601). Open both in the firewall for remote and clustered access. **Restart.**	Web Server
Web App Host	machine name	The machine name, IP, or DNS name remote machines use to reach the web app. An empty value falls back to the machine name. Set this before starting any load-balancer nodes.	Web Server
SSL / HTTPS	off	Enables HTTPS and the certificate actions. HTTPS is required for the Excel add-in and every Microsoft 365 integration. On a fresh Windows on-premises install it is turned on automatically at first boot. **Restart.**	Web Server
Repository Path	`<install>\Repository`	The root of the content and configuration store. Changing it affects all users. Relocate it safely with the Backup sequence below, never by moving the folder live. In a load-balanced deployment it must be a UNC network share reachable by every node. **Restart.**	Web Server
Recycle Service	off	An automatic daily application restart at a chosen time to free resources. Users refresh their browsers after the restart.	Web Server
Enable Daily Backup	on	An automatic daily Repository backup written as a restorable `.zip`.	Web Server, Backup
Backup Retention	30 days	How long a backup zip is kept before deletion.	Web Server, Backup
Parallel Job Execution	4 (range 1 to 50)	Server-level concurrency, the maximum number of top-level	Performance, Execution

Setting	Default	What it does	Tab
		jobs running at once. A load-balancer node's own settings screen carries a control of the same name that is a different setting, see the note below.	
Tasks per Job	3 (range 1 to 50)	Per-job concurrency, the maximum outputs or variations running within one job.	Performance, Execution
Enable Out of Process Job Execution	on	Isolates job execution from the web app process and frees memory after each run. Leave on unless support advises otherwise.	Performance, Execution
Default History Retention	30 days	How long job history is kept before it is purged.	Performance, Execution
Compress History Files	on	Compresses job-history databases at rest.	Performance, Execution
Notification Email	blank	The address that receives job notifications.	Performance, Execution
Send notification on failures only	off	Restricts job notifications to failures.	Performance, Execution
Custom Fonts Folder	blank	A folder of fonts to make available to report rendering.	Performance, Execution
Enable Out of Process Contribution Sessions	on	Runs contribution form calculation in separate worker processes.	Performance, Contribution
Contribution Log Level	Warning	The log level passed to contribution worker processes.	Performance, Contribution
Warm Worker Pool Size	0 (range 0 to 50)	How many contribution workers are kept started and waiting. 0 means no warm pool, so every session pays a cold start.	Performance, Contribution
Max Sessions per Worker	1 (range 1 to 50)	How many contribution sessions one worker process hosts. At the default, every live form is its own process.	Performance, Contribution

Setting	Default	What it does	Tab
Recycle Worker After Sessions	25	Recycle a contribution worker after it has served this many sessions. 0 never recycles.	Performance, Contribution
Warm Worker Idle Timeout	30 minutes	How long an idle warm contribution worker is kept before it exits.	Performance, Contribution
Enable Out of Process Reporting Queries	on	Runs an analyst's interactive Excel add-in query in a dedicated worker rather than inline on their web session. Turn it off to fall back to in-process execution.	Performance, Reporting
Reporting Log Level	Warning	The log level passed to the reporting query worker.	Performance, Reporting
Worker Start Timeout	90 seconds	How long to wait for a reporting worker to load its plugins and connect before failing the query.	Performance, Reporting
Query Timeout	300 seconds	How long to wait for a single interactive query to return.	Performance, Reporting
App-log retention	90 days	How long application logs are kept.	Logging
Debug Logging	off	A live instance-wide switch that raises the log level. It generates large logs and HTTP-Archive (HAR) files, turn it on only when support asks, and off again after the capture.	Logging
Audit retention	90 days	How long each audit log is kept. A value of 0 keeps them indefinitely.	Logging, Audit Logging

Size parallel execution against RAM, not just CPU

Two Performance settings set concurrency: **Parallel Job Execution** (top-level jobs at once) and **Tasks per Job** (variations within a job). The theoretical maximum concurrency is their product, for example 5 by 5 is 25 concurrent tasks.

> **Two different controls are called "Parallel Job Execution".** This one, on Settings ▸ Configuration ▸ Performance ▸ Execution, is **this server's own** job concurrency and accepts **1 to 50**. A load-balancer node's own settings screen, and the **max-jobs** spinner on the main server's Load Balancer grid, carry a control of the same name that is **that node's routing capacity** and accepts **0 to 128**. They are stored separately and do different work, so check which screen you are on before changing one.

The rule that matters for capacity planning: **peak memory scales with the job count.** If five jobs run in parallel and each needs roughly 2 GB, the instance needs about 10 GB free at the peak. Size the Performance tab against the RAM the server actually has, not just its CPU count. An under-sized box will run out of memory under load rather than simply running slowly. See System requirements for the baseline figures.

Capacity with a load balancer

In a load-balanced deployment the concurrency limits apply per server. Each server runs jobs up to its own **Parallel Job Execution** and **Tasks per Job**, so the total capacity of the instance is the per-node figure multiplied by the number of nodes. Size the limits against the RAM and CPU of a single node, then scale total throughput by adding nodes rather than by raising the limits beyond what one node can hold. See Deployment topology for load-balancer setup.

Relocate the Repository safely

The **Repository Path** is the root of the content and configuration store, so changing it affects every user. Never move the folder while the service is running, that risks corrupting the store the same way a live migration does.

Relocate it with this sequence instead:

1. On the Web Server tab, under **Backup**, create a backup.
2. Change the **Repository Path** to the new location.
3. **Restart** the service.
4. Restore the backup into the new location.

This gives you a clean, restorable copy and never leaves the store half-moved. The same backup zip is restorable to any Reportworq instance, so it also covers migration to a new server, see Migrate to a new server.

On a load-balanced deployment the Repository must be a UNC network share, for example `\\fileserver\reportworq`, reachable by every node rather than a local disk

path, because all nodes read and write the one shared store. Set the path to the share before starting the additional nodes.

SSL and self-signed certificates

HTTPS is mandatory for the Excel add-in and for every Microsoft 365 feature (Entra sign-in, SharePoint, Microsoft 365 email, and Graph). On a fresh Windows on-premises install, Reportworq turns on HTTPS for you at first boot with an auto-trusted self-signed certificate, so the first browse to the server is `https://` with no manual step (see Deployment topology). To turn on HTTPS yourself, or to replace the certificate, use the certificate actions on the **Web Server** tab. A self-signed certificate is enough to unblock the add-in and Microsoft 365, but replace it with a certificate from a trusted authority as soon as you can.

Replace or regenerate the self-signed certificate

Reportworq already creates and installs a self-signed certificate for you on a fresh install, so you use this action to **replace or regenerate** it, for example to cover additional host names or after a server rename. The **Create Self Signed Certificate** action generates a new certificate and turns on HTTPS if it is off. It prompts for one field:

- **Certificate Names.** Enter the names the certificate should cover, separated by commas, for example `localhost, <machine>, <machine>.<domain>`. On a domain-joined server the field is pre-filled with the machine name and its fully qualified domain name. Include every name users will use to reach the server: a name the certificate does not cover produces a certificate warning in the browser.

On create, Reportworq installs the certificate into the server's Trusted Root store, so the server itself trusts it with no manual import. The certificate password is generated and held internally, there is nothing to copy down and nothing is shown.

Let client machines trust the certificate

A self-signed certificate is trusted on the server automatically, but other machines' browsers do not trust it until you give them the public certificate.

1. On the **Web Server** tab, select **Download Trusted Root Cert (for client machines)** to export the public certificate file, `Reportworq-TrustedRoot.cer`. This action is available once a self-signed certificate exists.
2. Distribute that `.cer` file to each end-user workstation and import it into the machine's Trusted Root certification authorities, for example by Group Policy across the domain.

A certificate from a trusted public authority needs none of this client-side distribution, which is why it is recommended for anything beyond a quick test.

Install a certificate from a trusted authority

A certificate from a public or internal authority is uploaded into the Repository, not typed as a path.

1. On the **Web Server** tab, at **SSL Certificate**, select **Upload Certificate** and choose your `.pfx` (or `.p12`) file.
2. If the file is password-protected, select **Set Certificate Password** and enter it.
3. **Restart** the service, then browse the server over `https://` to confirm.

Once a certificate is installed, the **SSL Certificate** row names it and when it expires, so you can check what the instance is actually serving without going to the file system. Where a public `.cer` companion exists, that name is also the link that downloads it for client machines.

Certificate notes

- **The certificate names must match the URL users type.** Do not rely on `localhost` for remote users. If the certificate does not name the host users reach the server by, they get certificate warnings. This is the same rule as the Microsoft 365 advertised-address requirement, and the multi-name **Certificate Names** field lets one certificate cover every name in use.
- **There is no certificate-path field to fill in.** Earlier releases had a free-text path box, which was the most common cause of "HTTPS will not start" and could not work at all on a container deployment. Upload the certificate instead, and Reportworq stores it in the Repository, where every node and every restart finds it.
- **Behind a TLS-terminating reverse proxy there is no certificate to configure.** When the instance is served over HTTPS by a proxy, the SSL switch is locked on and the row says so. Do not try to bind a certificate as well.

To confirm a certificate's names against the URL you access, dump it on the server with `certutil -dump "cert.pfx"`.

Common tasks

- **Configure outbound email.** On the Email tab, select your provider, enter the mailbox and Default From address, save, then send a test email to confirm delivery before you schedule any distributions.
- **Tune for month-end.** Before the close, raise Parallel Job Execution and Tasks per Job on the Performance tab so a large batch runs within the window, sized against

available RAM per the rule above.

- **Set audit retention for compliance.** On the Logging tab, enable audit logging and set the retention to match your evidence-retention policy.
- **Provision a new workspace.** Workspaces are not configured on this screen. Create one and grant access to it on **Settings ▸ Security ▸ Workspaces**, see Workspaces.
- **Enable the Local API.** Regenerate a Local API key, enable the Local API, and save. A save with the API enabled but no key is rejected.

Notes and limits

- The **Authentication** settings are not on this screen. Configure them in The integrations hub.
- The **License** tab does not participate in the save flow. Its actions take effect immediately and activate against the external license service.
- The **Data Collection** tab is inert without the licensed **Contribution** capability (Data Collection is part of Contribution, not a separate line).
- The port, SSL, and Repository Path settings can break the instance if set carelessly. Change them deliberately and restart as prompted.

> **Going deeper.** For a one-click inventory of every job, report, parameter, burst set, schedule, and contact in the instance, generate the Repository metadata report from Settings, Auditing, System Log.

Performance and capacity planning

explanation administrator

Three questions arrive again and again, and they are the same question at three different moments:

- **Before you buy the servers.** How much hardware does this workload need?
- **While the work runs.** Is it going to finish inside the window, and is it spread across what we have?
- **After something got slow.** Why is it slow, and what do we change?

This section answers all three. Read this page first: it explains how Reportworq's capacity works, points at every measurement the product already keeps about itself, and sends you to the page that answers your question.

The one principle behind everything here. Reportworq measures its own work. Every run records where its time went and what it cost in memory and CPU, every contribution form can be profiled before anyone opens it, and a campaign can be load-tested before it goes live. So a capacity answer should almost always be a **measurement of your own workload**, not a rule of thumb about someone else's. The rules of thumb on these pages are starting points for the first estimate; the measurements are what you commit to.

Two capacity stories, not one

The single most useful thing to understand before sizing anything: **distribution and contribution consume the same server in different ways, hit different ceilings, and are measured with different tools.** Sizing one tells you almost nothing about the other, and a deployment that runs both needs both answers.

	Distribution (reports and jobs)	Contribution (input forms)
What consumes the server	Jobs running: generating, rendering, and distributing output	Forms being **held open** by contributors
What drives the cost	The number of jobs in flight, roughly 2 GB of RAM per concurrent job	Cells multiplied by concurrent sessions
The ceiling you hit	Memory exhaustion under parallel execution, or the window closing before the batch finishes	A hard **90% memory admission valve** that refuses new sessions, and the per-worker session cap
How it fails	Jobs run slowly, or the server runs out of memory under load	A contributor is told every server is at capacity and their form does not open
How you scale it	Concurrency limits per node, then more nodes	More memory or more servers, or a cheaper form
What you measure with	Run telemetry: **Run History** columns, the **Performance**, **Compare**, and **Timeline** tabs, and **Settings ▸ Performance ▸ Distribution**	**Campaign profiling**: memory profiling, stress testing, and analysis, plus the live **Settings ▸ Performance ▸ Contribution** console
Where it is sized	Distribute the load	Scale and size a contribution campaign

They also compete. Report execution and contribution sessions take memory from the same machine at the same time, so an instance doing both needs headroom for the busiest moment of each, not for their averages.

Start here

Your question	Go to
How many servers, and how big, for this workload?	Distribute the load
The batch does not finish inside our close window	Distribute the load, then Manage and improve performance
We added nodes and throughput barely moved	Distribute the load, the job-granularity rule
One report used to take three minutes and now takes twenty	Manage and improve performance
Everything feels slower since the upgrade	Manage and improve performance, the release pivot
One node in the cluster looks slower than the others	Manage and improve performance, the node pivot
Contributors are told all servers are at capacity	Scale and size a contribution campaign
How big can a contribution form be?	Profile and stress-test a campaign
A job failed rather than ran slowly	Diagnose job failures
Run history and stored output are consuming disk	Output retention

What Reportworq already measures about itself

Nothing in this list has to be switched on, and none of it is an add-on. Knowing what exists is most of the work, because the common mistake is to answer a capacity question by argument when a measurement was already sitting there.

For reports and jobs

Measurement	Where it lives	The question it answers
Elapsed vs baseline, Trend, Peak mem columns	A report's **Run History**	Was this run normal for this report?
Performance tab	A run's diagnostics	Where did this run's time go, down to the output, workbook, processing phase, and calculation engine?
Compare tab	A run's diagnostics	At what point in this report's history did its typical run time step, and did volume, release, or node change with it?
Timeline tab	A run's diagnostics	What did each part of the run coincide with, on one clock, including memory and CPU?
Settings ▸ Performance ▸ Distribution	Settings	Across every report at once: which are regressing, did a release regress us, is a node slower?
Run estimate chip	Activity, and the activity pulse popover	Is the run in progress going to finish roughly when it usually does?

Every run also records peak and average working set and maximum CPU, which is the raw material for sizing an execution host. All of it is written per run and **ages out when retention removes the run**, so pin a run you intend to compare against later.

For contribution

Measurement	Where it lives	The question it answers
Memory profiling	Campaign profiling, on the Campaign Dashboard	What does one form cost, in memory and per cell, and what does each additional concurrent form cost?
Stress testing	Campaign profiling	At the concurrency we sized for, what do contributors actually experience?
Analysis	Campaign profiling	Merged across load generators: typical, most-users, and slowest timings, plus slow-tail and slowdown findings
Settings ▸ Performance ▸ Contribution	Settings	Right now: per-server memory, per-worker sessions, and how close the fleet is to the admission valve

Memory profiling is read-only and safe against a real campaign. Stress testing writes data and submits forms, so it runs only against a disposable copy.

The baseline the estimates start from

These are the published requirements, and they are where a first estimate begins before you measure anything.

Figure	Value
CPU	4-core, 2 GHz or faster, x64
RAM	32 GB minimum, plus roughly **2 GB per concurrent job** on top of the base
Disk	10 GB free, growing with the Repository, run history, and logs
Default concurrency	4 top-level jobs at once, 3 outputs within a job

The RAM figure is a stated requirement, not something the software enforces at startup, and an under-sized server runs out of memory under load rather than simply running slowly. Full detail, including the Linux fonts requirement and the two ports every server needs, is in System requirements.

Notes and limits

- **This is not alerting, and Reportworq has no performance thresholds to configure.** There are no notifications when a report regresses and no service-level targets to set. Every surface described here is something an administrator opens and reads. Build any alerting you need around the measurements rather than expecting the product to page you.
- **Comparison needs history.** A report needs at least 3 comparable earlier runs before any baseline comparison is offered, and 6 before the Compare tab will look for a step change. A freshly upgraded instance has a thin picture until new runs accumulate.
- **Measurements age out with the run.** Performance data is stored with the run and is removed when retention removes it. Pin the runs you expect to compare against, such as each period-close run. See Output retention.
- **Runs that predate performance telemetry never gain it.** Their columns report that they carry no measurement rather than reporting zero, which is a different fact and is displayed as one.
- **A view-only User sees none of the diagnostic detail.** The performance columns and the diagnostics tabs sit behind **View diagnostics**, which that role

does not have. Authors and administrators reach them.

> **Going deeper.** For sizing and spreading a workload, see Distribute the load. For diagnosing and improving content that is already slow, see Manage and improve performance. For the deployment shapes and the load-balanced cluster itself, see Deployment topology. For the settings that these pages tune, see Server configuration.

Data collection processing

how-to administrator

Data Collection is the write-back layer that takes collected contribution input, an uploaded input workbook or one received in a monitored mailbox, and writes its values back to the target system, for example a TM1 or Planning Analytics cube or a SQL table. This guide is for the administrator who runs and troubleshoots that processing.

Data Collection is part of the licensed **Contribution** capability. Without it, the Data Collection settings show "Data Collection is not included in your license," and you contact Reportworq to enable it.

When to use it. After contribution forms are submitted or approved and a campaign export runs, or whenever input workbooks arrive by upload or into the monitored mailbox and need to post to the source planning system.

Before you start

- You need the **Contribution** capability licensed (it includes Data Collection) and Administrator rights.
- A write-back target must be reachable: TM1 / Planning Analytics, SQL, Workday Adaptive, or Anaplan. Vena is a reporting source only and cannot be a write-back target.
- For the mailbox intake path, have a dedicated mailbox and its credentials ready. See the section below, **Set up the monitored mailbox**.
- Data Collection processes input workbooks. It is not the destination of a contribution campaign's Export action, which writes to the target system itself. See **Two write-back paths** below.

Two write-back paths

Reportworq writes contribution data back to a planning system in two ways. They share the licensed **Contribution** capability and nothing else: they are separate pipelines, they reach the target by different means, and a given round uses one or the other.

- **Write-back through Data Collection** (this guide). Input workbooks arrive by upload or in a monitored mailbox, and the Data Collection engine processes them. The values are posted by the **hidden write-back formulas inside the workbook** (`DBS`, `DBSW`, `DBSS`, `RWSQLUPDATE`, and the equivalents for other targets), which the engine evaluates in write-back mode.
- **Write-back through a contribution campaign.** Contributors enter values in a generated form, and the campaign holds them in its own runtime. Nothing reaches

the planning system until the campaign's **Export** action runs, on a trigger, on a schedule, or by hand. The export builds a table of the collected values and hands it to an export provider, which writes to the target directly. For that path, see Author a contribution campaign.

> **A campaign export does not go through the Data Collection engine.** It does not queue a write-back item, and it does not evaluate the hidden write-back formulas. If you are looking for a campaign's values in the write-back activity list, they will not be there; a campaign's exports are recorded in the campaign's own activity.

The rest of this guide covers the Data Collection engine path.

How write-back items are processed

Write-back items enter one of two ways and then move through the same processing pipeline.

1. **Upload.** An administrator or author uploads an input workbook from the write-back activity view (**Upload…**). With **Automatically process uploaded files** turned on, the upload processes on arrival. Otherwise, select the pending item and run **Process**.
2. **Mailbox.** The monitored inbox is polled on a configured frequency, and matching messages become items. Select **Check Inbox** to poll on demand.

Each item begins **Pending**, moves to **Running** as the engine writes its data to the target, and finishes **Complete**. The write-back activity grid lists every item with its **Filename**, **Submitter**, **Uploaded**, **Processed**, and **Status**, plus per-row **Run** and **Details** actions.

After a file is processed, an audit record of the write-back outcome is written to the input archive.

Configure Data Collection

Open **Configuration ▸ Data Collection** to set the processing behavior.

Setting	What it does
Enable Data Collection	Turns the feature on (requires the Contribution capability).
Automatically process uploaded files	Processes an upload immediately instead of waiting for a manual **Process**.
Enable Verbose Logging	Adds diagnostic detail for troubleshooting.
Enable Trace Logging	Adds deeper diagnostic detail. Use only when Reportworq support asks, since it is heavier.
Process Messages	How far back, in days, mailbox messages are considered (the lookback window).
Polling Frequency	How often the inbox is polled.
History Retention	How long, in days, processed history is retained.
Proportional Writeback (PA)	Off by default. Off rejects consolidated-cell updates; on lets Planning Analytics spread the value. See below.

Let consolidated-cell writes through (Proportional Spreading)

By default, Reportworq rejects any attempted update to a Planning Analytics consolidated cell, matching the Planning Analytics Excel add-in. If a form appears to "lose" a top-level input, that input was most likely rejected for this reason.

To allow a submission to a consolidated cell, turn on **Proportional Writeback**. The value is then sent to Planning Analytics with the proportional-spread marker (a `P` prefix), so Planning Analytics allocates it across the leaf cells based on their existing data.

Set up the monitored mailbox

Open **Configuration ▸ Data Collection ▸ Mailbox Settings** and complete the connection.

Setting	Notes
Provider	**Modern Authentication** (Microsoft 365 OAuth app registration) or **Basic Auth** (username and password).
Protocol	**POP3** or **IMAP**.
Server / Port	Your mail server host and port.
Use SSL	TLS/SSL for the connection.
Username / Mailbox address	The mailbox account (the mailbox address under Modern Authentication).
Password	The mailbox credential (Basic Auth).

Exchange Online mailbox: two traps

If the monitored mailbox never picks up messages against Exchange Online, it is almost always one of these two causes:

- **POP3 is disabled on the account.** Reportworq polls the monitored mailbox over POP3. Exchange Online supports POP3 but it is disabled by default and must be explicitly enabled on the specific mailbox account, per account, in the Exchange admin center.
- **Basic Auth is deprecated.** Microsoft has retired Basic Auth for POP3 and IMAP in Exchange Online, so the Basic Auth mailbox option fails against it. Configure **Modern Authentication** / **OAuth2** instead, using the Microsoft 365 OAuth app registration. See Microsoft 365 OAuth setup.

Prune the monitored mailbox

Reportworq records processed mailbox messages in its own internal database so it never reprocesses the same message. That means the mailbox itself is not emptied by processing, and processed messages accumulate in it over time.

Apply a retention rule to the monitored inbox, for example archive or delete messages after a set period, so the mailbox does not grow unbounded. The processed-message history that Reportworq keeps is separately bounded by the **Days to Keep** setting.

Notes and limits

- Data Collection is unavailable without the licensed **Contribution** capability.
- Consolidated-cell writes are silently rejected unless Proportional Writeback is on. This is the first thing to check when a top-level submission does not land.

- Write-back targets are TM1 / Planning Analytics, SQL, Workday Adaptive, and Anaplan. Vena is a reporting source only and cannot be a write-back target.
- The write-back input formulas that make a form post its data (DBS/DBSW for Planning Analytics, RWSQLUPDATE/RWSQLUPSERT for SQL, AIINPUT/AIMODELEDINPUT for Workday Adaptive, and ANAPLANINPUT for Anaplan) live in hidden rows by design and render as `#NAME` in the contributor's view. Do not un-hide them.

> **Going deeper.** For live memory pressure on the contribution worker fleet during a large collection round, see Contribution processes. For symptom-to-fix pairs, see Troubleshooting index.

Audit logs

reference administrator

Settings ▸ Auditing is the read-only surface for reviewing what happened on the instance. It presents four logs as tabs, in this order: **System Log**, **Job Log**, **AI Interactions**, and **MCP Server Log**. It is the instance's accountability layer, the authoritative record for compliance, security review, AI cost attribution, and execution troubleshooting.

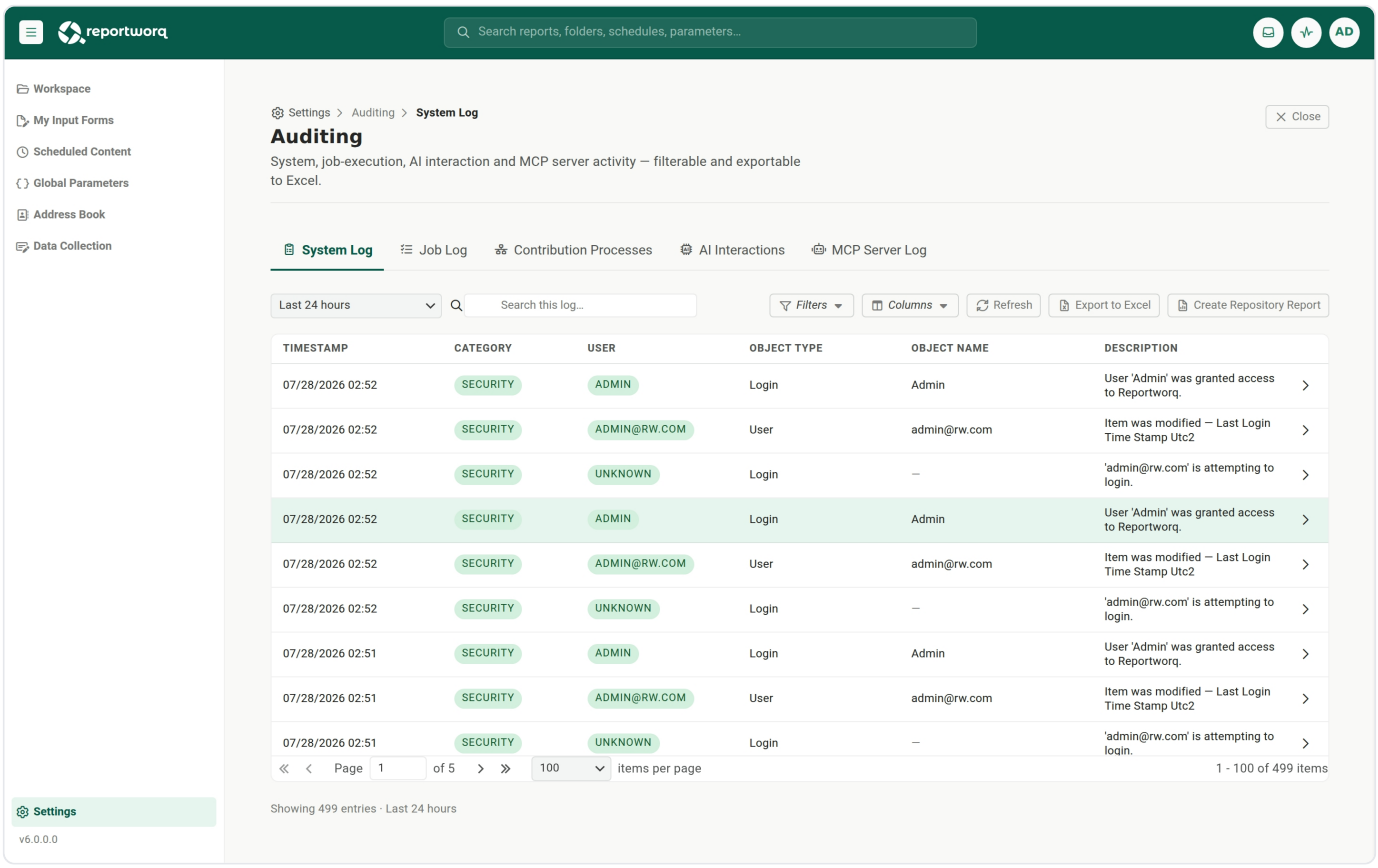
Auditing is administrator-only; a non-administrator never sees it. The System and Job logs are governed by the retention and enablement settings on **Configuration ▸ Logging**, described in Server configuration.

The four tabs

Tab	What it records
System Log	Administrative and operational events: configuration changes, security events, scheduling actions, job authoring, content changes, licensing, and version events.
Job Log	The outcome of every job execution, one immutable record per run.
AI Interactions	Every call to a configured AI provider, the AI usage and token-billing trail.
MCP Server Log	MCP request activity, the desktop-AI and Copilot access path.

System Log is active when the screen first opens. Drilling into any row hides the tab strip and shows a detail surface with a **Close** that returns to the list.

System Log



The System Log is the authoritative record of administrative and operational events. It records application actions on logical items (one entry per item change, not per physical file write), with a property-level before-and-after difference on updates, and bulk entries that summarize a multi-item operation (for example "7 item(s) deleted") with per-item detail embedded. Ids are resolved to names at write time, so descriptions carry no raw ids.

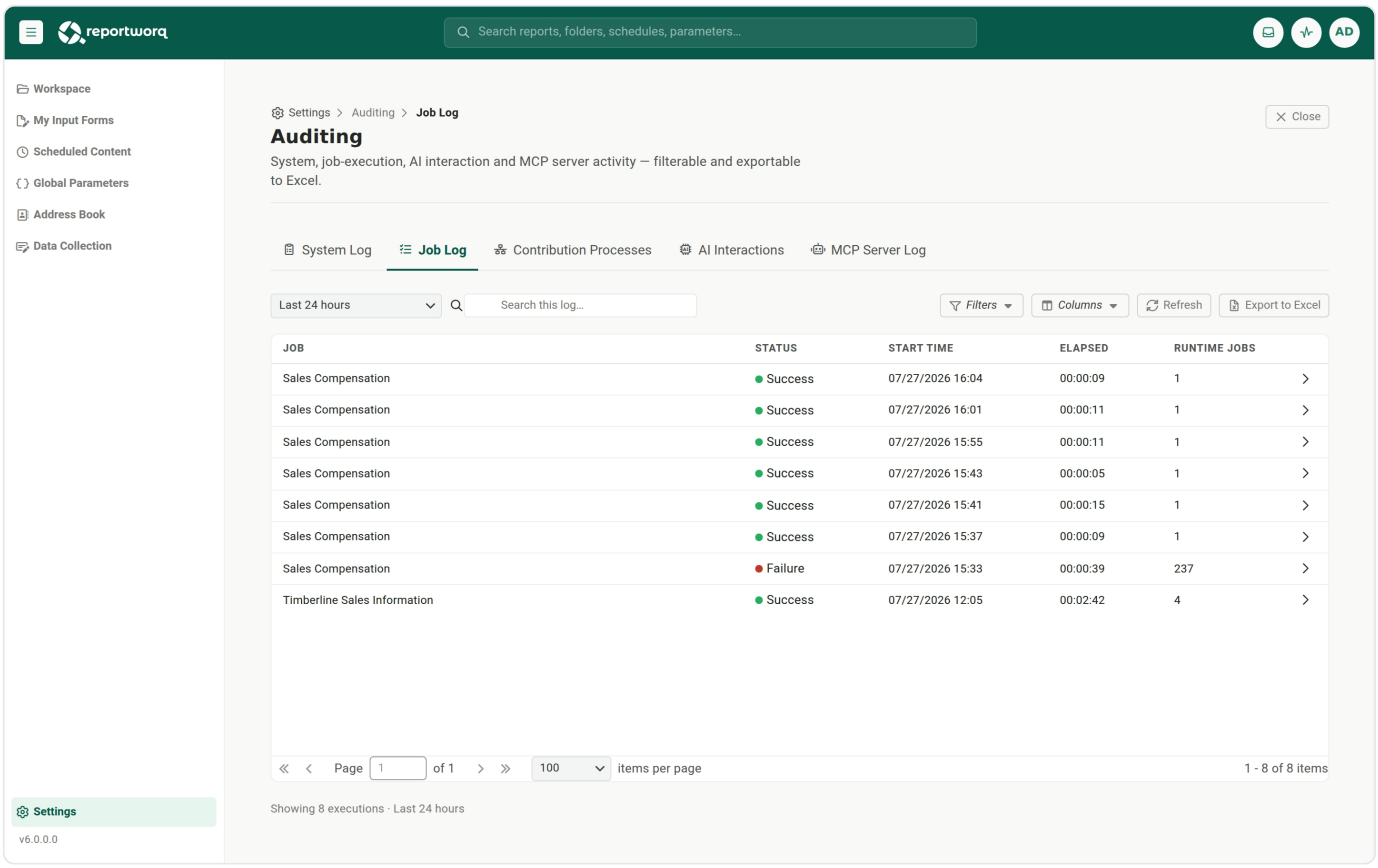
Column	Meaning
Source	The machine and component, rendered as `{MachineName} - {Application}`.
Timestamp	When the event occurred.
Category	The object category (see below).
User	The acting account; system actions show as **SYSTEM**.
Object Type / Object Name	What was acted on.
File	The associated file, where applicable.
Modifications	The change detail; an update carries the before-and-after difference.
Description	A human-readable summary.

Entry types are `INFORMATION`, `WARNING`, `ERROR`, `CREATE`, `UPDATE`, and `DELETE`. Each entry also carries an environment block: **MachineName** (the host), **Application** (the emitting component: Web App, Job Runner, Staff Console, or Load Balancer), and **HostingType** (OnPrem, SingleTenantCloud, or MultiTenantCloud).

The category comes from a fixed set: Informational (app start and stop, software updates), Licensing, Security (user management and sign-in), Settings (datasource connections, providers, distributors), BuiltInReportProvider, Scheduling, Jobs, Contacts, GlobalVariables, JobHistory, Writeback, MessageTemplates, ReportDataModels, ReportModels, DataModels, ReportSpecs, GlobalParameterValues, TempStorage, DataModelContainers, and WorkspaceFiles.

A **Create Repository Report** action is available on this tab for a full repository snapshot. See The repository metadata report.

Job Log



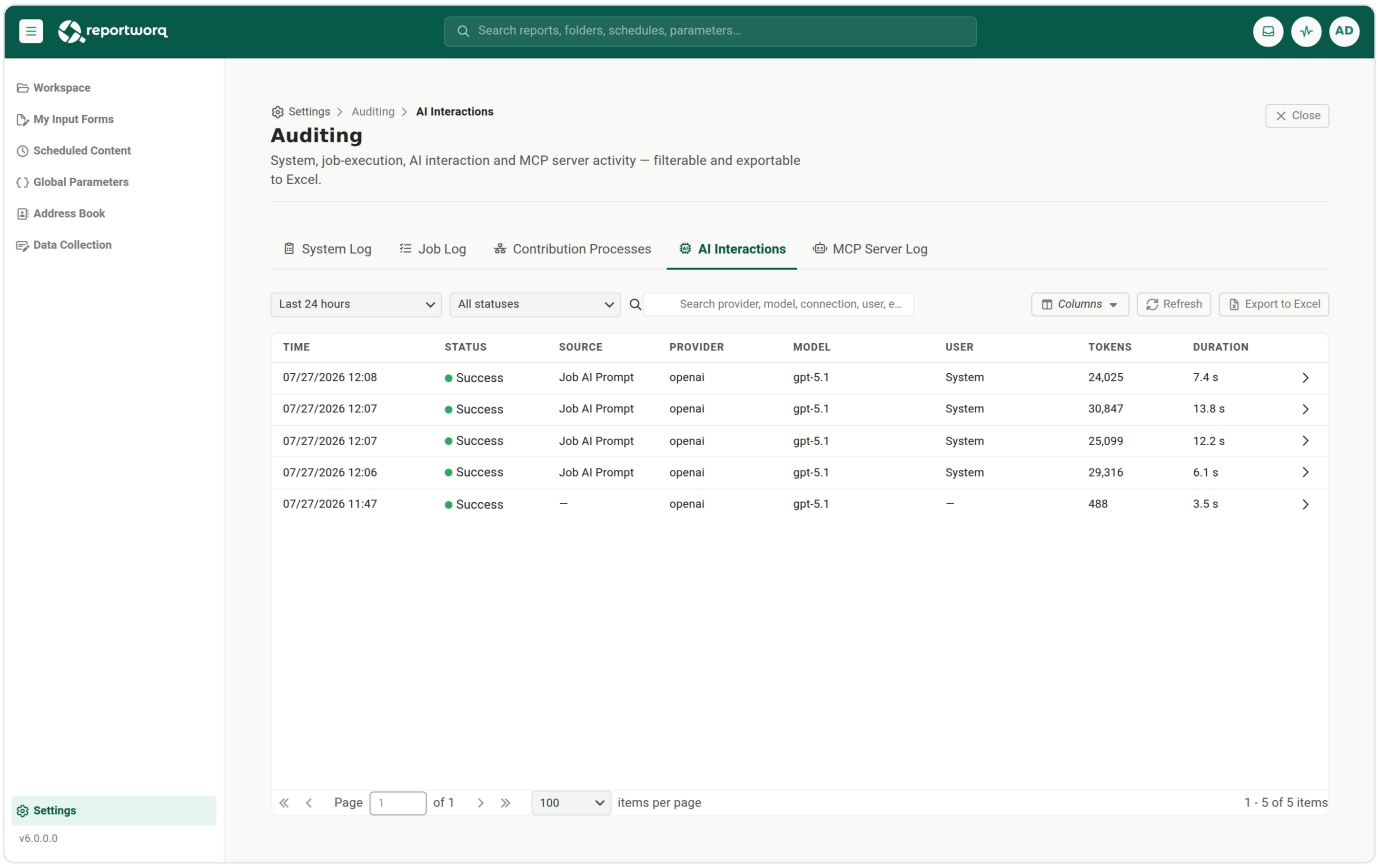
The Job Log records the outcome of every job execution. Each execution is written once, as an immutable record keyed by its execution id, on job completion, so a systems integration can safely consume one record per run.

Column	Meaning
Name	The job name.
Status	The execution outcome.
Execution ID	The unique key for the run.
Start / End / Elapsed Time	Timing.
Parameter Overrides	Any parameter values supplied for the run.
RuntimeJob Count / Runtime Job Names	The runtime jobs the execution produced.

A row expands to a nested grid of its runtime jobs, each with name, status, elapsed time, and parameters. The record also carries the machine name, the OS service account the run executed under, and the workspace.

> **Retention is judged on the record's creation date, and an integration must consume before the window expires.** A Job History record is written once and never updated, but retention deletes it based on when it was created, not last touched. If an external system consumes Job History as its system of record, set that log's retention to `0` (keep indefinitely) so records are not swept out from under it. Configure this on Configuration ▸ Logging.

AI Interactions



The AI Interactions log records every call to a configured AI provider (for example OpenAI, Azure OpenAI, or AWS Bedrock). It carries timestamp and status, the source, the provider, model, and connection, the user, workspace, and job id, the request id, token counts (prompt, completion, and total), prompt and response sizes, duration, and the finish reason or error. Prompt and response bodies are stored compressed and shown on drill-in. It is the AI usage and token-billing trail.

MCP Server Log

The MCP Server Log is Reportworq's audit trail of AI access to reports over MCP, the access path used by desktop AI clients and the Copilot integration. It records every MCP tool call that reached a tool (who saw what, and when) and every request rejected at the

authentication gate before any tool ran. It audits the tool request-and-result exchange, not the AI model's conversation; that conversation is recorded in the AI Interactions log.

Each event captures the time, the outcome (Success, Tool error, Not found, Access denied, or Gate denied), the tool called, the resolved target report and workspace, the caller, the client (derived from the User-Agent, for example Copilot or an Anthropic client), the transport (direct or CloudHub relay), and the duration. Drilling into a record shows the full request arguments and result body. Credentials are never written to a record.

MCP audit logging is on by default, set at **Settings ▸ Configuration ▸ Audit Logging**, with a 90-day retention window. Turning it off stops new records but leaves existing ones in place. For the full field list, outcome definitions, and toolbar detail, see MCP audit.

Controls on every tab

Each tab shares a common toolbar:

- A **time range** dropdown (Last hour through All time).
- A **search** box, plus **Filters** and **Columns** controls.
- **Refresh**.
- **Export to Excel**, which downloads a per-log workbook (an empty log still produces a header-only workbook). A drilled-in record can be exported with **Download record**.

Notes and limits

- The logs are a durable, read-only record, not a live monitor of running jobs. For live and near-term job activity, use Activity and Run History.
- Records are subject to retention, so export before the window expires. On the retention settings, `0` means keep indefinitely, and retention is judged on the record's creation date.
- The AI Interactions and MCP tabs are empty on an instance that has made no such calls.
- **Contribution activity is not audited.** Opening, editing, submitting, approving, or rejecting an input form leaves no audit record here or anywhere else. What a contribution campaign collected is visible in the campaign itself, not in these logs.

> **Going deeper.** To bundle log files into a support packet (a different tool from this viewer), see Log diagnostics and support packets.

Deployment topology

explanation administrator

Reportworq runs in more than one shape, and for high throughput an on-premises instance can be scaled out across several servers. Read this page first, before you install: it helps you choose a deployment, summarizes what each one needs, and explains how a load-balanced cluster is put together, including the handful of network and configuration facts that make the difference between a cluster that forms and one that silently does not.

Choose your deployment

Deployment	When to use it	What it needs
On-premises (Windows)	The standard install. You host Reportworq on your own Windows servers.	Windows Server 2016 or later, the MSI installer, and the Reportworq 6 Bootstrap Service. See System requirements.
On-premises (Linux)	You standardize on Linux hosts and do not need the Windows installer.	A current 64-bit Linux distribution such as Ubuntu, and the tarball. No installer, no Windows service.
Single-tenant cloud (Azure container)	You want Reportworq hosted in Azure with a disposable, image-based runtime.	An Azure App Service or Container Apps host, a repository on an Azure Files mount, and sign-in behind Entra / OIDC.
CloudHub	You want external AI clients to reach an on-premises instance without exposing its API surface.	An existing on-premises or cloud instance, registered with CloudHub. CloudHub is a relay, not a place Reportworq is hosted.

The full hardware, browser, and network requirements are the same across shapes and live on one page, see System requirements.

Whichever shape you choose, **Reportworq 5 and Reportworq 6 can run side by side on the same server**. Version 6 installs into its own folder and listens on its own default port, so it does not disturb a running version 5, which keeps its own folder and port. That makes a staged migration practical: stand version 6 up alongside version 5, bring content across, and validate before you decommission anything.

On-premises (Windows)

The customer hosts and runs Reportworq on their own Windows servers and activates against the licensing service. The MSI installer registers the **Reportworq 6 Bootstrap**

Service, which hosts the web application and the job runtime. The default web port is **8600** (with the Real-time Event Hub on **8601**), and a fresh install serves over HTTPS out of the box. See Install Reportworq.

On-premises (Linux)

The same application ships as a Linux tarball that runs the web binary directly on .NET 10. There is no MSI and no Windows service. Because the automatic first-boot certificate is Windows-only, enable HTTPS on Linux through configuration or a TLS-terminating reverse proxy (see below).

Single-tenant cloud (Azure container)

The runtime tier is platform-neutral, so a single-tenant instance can be containerized and run on Azure App Service or Container Apps:

- The application runs from a read-only container image and takes its boot settings (repository path, port, HTTPS, instance identity) from the environment rather than a writable settings file.
- The **Repository lives on an Azure Files mount**; that mounted storage is the only durable state. Azure Files is the supported storage today. An Azure Blob storage backend is planned for a future release.
- Sign-in runs behind **Entra / OIDC**, and stored credentials are portable, so a lift-and-shift keeps them without re-entry.
- Because the image is read-only, you update by **swapping the container image tag or the App Service container setting and restarting**, not with the in-product updater.
- One **web instance** owns the UI and the scheduler; you scale throughput by adding **worker nodes** against the same shared repository. The web tier itself is not horizontally scaled today.

Sizing and moving to Azure

Moving an existing on-premises instance into an Azure container is the change customers most often ask for help with, so this section is deliberately concrete.

Sizing the App Service plan. The hardware guidance is the same as any other deployment: allow for the base footprint plus roughly **2 GB of RAM per concurrent job**, see System requirements. Translating that to a plan:

Plan	Use it for
B2	A proof of concept or a pilot. Enough to stand the product up and run a light workload; not a production footprint.
P1v3 or above	Production. Step up to this for the RAM and throughput a real job load needs.

Size the plan against your **peak concurrency**, not your average. Peak memory tracks the number of jobs in flight, so the plan has to hold the busiest moment rather than the typical one, and the **Parallel Job Execution** limit is what you use to keep that ceiling predictable, see Server configuration.

What the deployment consists of. An App Service plan and App Service running the container image, a storage account with an **Azure Files** share for the repository, and log storage. The Files share is the only durable state, so it is the thing to back up and the thing to size for growth.

What to expect when you move. Three differences catch people who are used to the Windows install:

- **Stored credentials travel.** Secrets are encrypted with a portable key rather than tied to a Windows machine, so a lift and shift keeps them without re-entry.
- **Configuration moves from a settings file to the environment.** The container will not rewrite its own settings, so the repository path, port, HTTPS and instance identity all arrive as environment variables.
- **Certificates and custom fonts come from the repository**, not from the host filesystem, because the image is read only.

Two things you cannot do on this path, both worth knowing before you commit to it: the in-product updater is disabled, so you update by swapping the image tag, see Update Reportworq; and a **v5 repository import is not available**, so migrate a v5 repository with a Windows or Linux install first and point the container at the resulting v6 repository.

Scaling. Run exactly **one web instance**, which owns the UI and the scheduler, and add **worker nodes** against the same repository for throughput. The web tier is not horizontally scaled today.

For the step-by-step install, see Install Reportworq.

CloudHub

CloudHub is a **Reportworq-operated relay**, not a hosting tier. It lets external clients reach an on-premises instance without exposing its direct API surface. Two kinds of client

use it:

- **AI clients**, such as Copilot and desktop MCP clients, reaching your curated reports.
- **External services calling the REST API**, so a system outside your network can start a Reportworq job. This is how integration and automation platforms such as Workato, and other cloud-hosted services, trigger reporting without a VPN or an inbound firewall rule into your network.

Registering an on-premises instance with CloudHub is itself the approval for that channel, and it is not coupled to the Local API switch: locking down the direct API does not disable CloudHub-relayed access, and vice versa. Keep the two mental models distinct.

Calling the REST API through CloudHub still requires the **API** capability on your license, see Licensing and entitlements.

HTTPS, first boot, and reverse proxies

Reportworq serves over HTTPS, which the Excel add-in and every Microsoft 365 integration require. How HTTPS comes on depends on the deployment shape.

- **HTTPS out of the box.** A fresh Windows on-premises install generates a self-signed certificate, trusts it on the server, and turns on HTTPS at first boot, so the first browse to the server is `https://`. This applies only to a genuinely new Windows on-premises install: it never changes an existing HTTP deployment, a load-balancer node, or a Linux or container host. For the certificate actions, and how to let client machines trust the certificate, see Server configuration.
- **Behind a reverse proxy.** When Reportworq runs behind a TLS-terminating reverse proxy (such as Caddy, nginx, or IIS ARR) that forwards plain HTTP to it, set **`BehindReverseProxy`** to `true` in `settings.json` and restart. There is no screen for this setting, so hand-edit the file. With it set, Reportworq honors the `X-Forwarded-Proto` header and builds its absolute links, including OAuth redirect URIs and Office add-in callbacks, with the correct `https` scheme instead of `http`.
- **HTTP to HTTPS redirect.** When HTTPS is on and port 80 is free, Reportworq also listens on port 80 and redirects those requests to HTTPS.

Scaling out with a load-balanced cluster

When a single server cannot finish the job load inside its window, an on-premises instance can be scaled out into a **load-balanced cluster**. The nodes share one Repository, and the load balancer **distributes whole jobs** across them: **each job runs atomically on a single node**. A node that is not allowed to run a queued job re-queues it so another node picks it up. This spreads batch job execution, Excel reporting, and

campaign-form load across the cluster.

Settings, Load Balancer is the screen that monitors and manages the nodes. It shows a grid with, per node, an enable switch, a max-jobs capacity spinner, and row actions **Rename**, **Filter jobs**, **Restart**, and **Remove**.

Load balancing is licensed as the **Load Balancer** node count, enabled per license on request; a count of 0 means it is off. A single-node install shows an empty node grid.

How a node joins the cluster

A node joins through the **setup wizard**, not a button on the Load Balancer screen. On each new server:

1. Install Reportworq normally, see Install Reportworq.
2. Set the **Reportworq 6 Bootstrap Service Log On As** account to one that can reach the shared Repository and the source network folders.
3. In the setup wizard's **Repository** step, choose **Add a load balancer to an existing web app** and enter the **UNC path to the shared Repository**. (On a server whose Repository is already configured, reopen the wizard from **Settings > Configuration > Web Server > Reconfigure** and choose the same option.)
4. The node applies the choice, restarts into execution mode, and joins the cluster. No further node-side configuration is needed.

If a node does not join, the usual cause is that its Log On As account cannot reach the shared Repository UNC path.

Network prerequisites

Two ports must be open in the firewall on the main server **and on every load-balancer node**:

Port	Purpose
8600 (default)	The main web server.
8601 (default)	The Real-time Event Hub bus, which is always the web port plus one.

Opening only the web port is the classic mistake. The cluster appears to connect, but its real-time coordination fails quietly because the Event Hub bus is blocked. Open both on every server.

The **Repository must be a UNC network share** reachable by every node, and any source report providers (especially network-folder providers) must be reachable by every node too. Finally, set **Web App Host** in the main server's Server configuration to a machine name, IP, or DNS name the nodes can resolve, **before** you start the load

balancers. If it is left to coalesce to a bare machine name the nodes cannot resolve, they will not join.

Sizing per-node capacity and concurrency

A load-balancer node, reached at its own address on port 8600, presents a reduced shell whose **Execution** category sets its own behavior: a **Server Instance Name** that identifies it in job screens, the Load Balancer monitor, and the node pivot on Settings ▸ Performance; an **Enable Execution** checkbox; and a **Parallel Job Execution** value from **0 to 128**. These are the same three values as that node's row on the main server's Load Balancer grid, where they appear as the node name, the enable switch, and the **max-jobs** spinner, so you can edit them from either end.

Two controls share the name "Parallel Job Execution", and they are different settings.

Where	What it governs	Range
A node's own screen, or the max-jobs spinner on the Load Balancer grid	How many jobs the load balancer routes to that node	0 to 128
Settings ▸ Configuration ▸ Performance ▸ Execution	How many top-level jobs that server runs at once	1 to 50

Size them together. A node's max-jobs value should not exceed what its RAM can actually hold, since peak memory scales with the number of jobs in flight and the routing limit knows nothing about the machine's memory. See Server configuration for the concurrency settings.

Concentrating execution on dedicated nodes. A node's **Parallel Job Execution** accepts **0**, and its **Enable Execution** checkbox can be cleared, so a node can stay registered in the cluster and take no work. That is how you drain a node before maintenance without removing it and having it re-register.

The main server's own **Parallel Job Execution**, on Settings ▸ Configuration ▸ Performance, accepts **1 to 50** and cannot be set to 0, so the main server always keeps some execution capacity of its own. To concentrate batch work on dedicated nodes, lower the main server's value and raise the nodes', rather than trying to zero the main server. Because interactive Excel add-in refresh is not distributed and runs on whichever node holds the analyst's session, leaving the main server a small amount of capacity is usually the split you want anyway.

Routing jobs to specific nodes with Filter jobs

The per-node **Filter jobs** action opens the **Filter Jobs** dialog, where you decide which jobs a node is allowed to run. The dialog is a tri-state (include, exclude, or inherit) tree over the instance's workspaces, folders, and jobs, with two modes:

- **Include all jobs and workspaces** (the default), so the node runs everything.
- **Include or exclude specified jobs and workspaces**, so you can pin certain jobs to certain nodes.

Reportworq saves the filter on the node and enforces it when routing: for each queued job it checks the job, then walks up its parent folders, then its workspace, and re-queues any job the node is not allowed to run so another node picks it up. By default every node runs everything.

Reading the Load Balancer monitor

- A node's **Status** reflects whether it made a successful job-dequeue attempt in the last two minutes.
- All servers in a cluster should run the same Reportworq version.
- **Restart** takes a minute or two to complete.
- **Remove** deletes a node entry, but a still-running server reappears, because nodes self-register autonomously. To retire a node for good, stop its service.

Notes and limits

- Node management applies only to a real multi-node cluster. A single-node install shows the empty grid "No load balancer servers are registered."
- Load balancing requires a **Load Balancer** node count of 1 or more on your license; a count of 0 disables it. See Licensing and entitlements.
- Interactive Excel add-in refresh is **not yet distributed** across cluster nodes. An analyst's live refresh runs on whichever node holds their session, regardless of the load balancer's capacity settings. Distributing interactive refresh across nodes is planned for a later phase; today the cluster distributes batch job execution.

> **Going deeper.** For the per-node install and service-account setup, see Install Reportworq. For the concurrency settings that pair with per-node capacity, see Server configuration. For sizing a cluster against a workload, and why job granularity decides whether extra nodes help, see Distribute the load.

Install Reportworq

how-to administrator

Reportworq installs three ways, and which one you use is a consequence of the deployment you chose: **Windows MSI**, **Linux tarball**, or a **Docker / Azure App Service container**. **Decide the deployment first**, see Deployment topology, then follow the matching path below.

You run the installer, set the service account that jobs run under, then finish setup and activate the license in the browser. The steps below cover the install, the service account, product activation, and how to uninstall without stranding a license or losing reusable state.

Before you begin

- A supported server, see System requirements.
- Administrator rights on the server.
- Your Reportworq license key.
- For online activation, outbound connectivity from the server to the licensing service. For an air-gapped server, use the manual activation path below.
- If jobs will read from or write to network shares, a domain account to run the service under, see Set the Bootstrap service account.

Choose how you install

Platform	How it installs	Where to go
Windows Server	The MSI registers the **Reportworq 6 Bootstrap Service**, which hosts the web application and the job runtime. Default port **8600**.	Install on Windows
Linux	A tarball with an `install.sh` that creates a service user and a systemd unit. No installer UI, no Windows service. Default port **8080**.	Install on Linux
Docker or Azure App Service	A read-only container image configured entirely from environment variables. Nothing is installed on a host.	Install as a container

The three differ in more than mechanics, so do not carry assumptions across:

- **The default port differs.** Windows listens on **8600**, Linux and containers on **8080**. The Real-time Event Hub is always the web port **plus one**, and both must be open.

- **HTTPS arrives differently.** A fresh Windows install generates a self-signed certificate and serves HTTPS at first boot. Linux and containers do **not**: they serve HTTP until you configure a certificate or terminate TLS at a reverse proxy. HTTPS is mandatory for the Excel add-in and every Microsoft 365 feature, so this is not optional in practice.
- **Updates differ.** Windows and Linux update in product. A container is updated by swapping its image, see Update Reportworq.
- **Hardware requirements do not differ.** Use the same figures for all three, see System requirements.

Run the installer (Windows)

1. Run the Reportworq installer (`Reportworq_Installer.exe`) on the server.
2. Accept the license agreement. Acceptance is mandatory, the installer will not continue without it.
3. Let the installer complete. It installs to the default path `C:\Program Files\Reportworq 6` and registers and starts the **Reportworq 6 Bootstrap Service**, the Windows service that hosts the web application and the job runtime.

Reportworq 6 installs alongside an existing Reportworq 5 on the same server. The two use a different install folder, a different Windows service, and a different default port (8600 for v6, 8300 for v5), so they run side by side without conflict.

Install on Linux

1. **Prepare the host.** A 64-bit Linux distribution with **glibc** and **systemd** (Debian/Ubuntu, RHEL/Rocky/Alma, or SUSE). You need root or sudo.
2. **Install the native prerequisites**, which the render stack depends on: `fontconfig`, `libfontconfig1`, `libfreetype6`, and `libgomp1`.
3. **Install at least one font package**, for example `fonts-liberation` or `fonts-dejavu`, or your corporate fonts. This is **not optional**: a clean Linux server has almost no fonts, and report output renders blank or boxed without them.
4. **Unpack the tarball and run its `install.sh`.** It creates a `reportworq` service user and registers the systemd unit. The application lands in `/opt/reportworq`, and the repository defaults to `/var/lib/reportworq/repository`.
5. **Set configuration** in `/etc/reportworq/reportworq.env`. The web port is `RW_PORT`, default **8080**.
6. **Open the firewall** for the web port and the Event Hub port (`RW_PORT` plus one).
7. **Start the service**, then browse to the server on its port to reach the EULA and the first-run wizard, see First-run setup.

Logs are written to `/var/log/reportworq`. Because the Windows first-boot certificate generator does not apply here, configure TLS before you rely on the Excel add-in or any

Microsoft 365 feature, see Deployment topology.

Install as a container

There is nothing to install on a host. The image carries the application, and every setting arrives through the environment.

1. **Provide durable storage for the repository.** On Azure, that is an **Azure Files** mount. This mounted storage is the only state that survives the container, so it is the thing to back up.
2. **Set the environment**: the repository path, the web port (`RW_PORT`), HTTPS, and the instance identity. The image already runs under the immutable deployment profile, so it will not try to rewrite its own settings file.
3. **Provide the certificate and any custom fonts through the repository**, since the container filesystem is read only.
4. **Start the container** and browse to it to reach first-run setup.

Two consequences of the read-only image are worth knowing up front: **the in-product updater is disabled** (you update by swapping the image tag, see Update Reportworq), and **a v5 repository import is not available** on this path. Import a v5 repository with a Windows or Linux install first, then point the container at the resulting v6 repository.

The web tier runs as a single instance; scale throughput by adding worker nodes against the same repository. See Deployment topology.

Set the Bootstrap service account

The Bootstrap service runs as **Local System** by default. Local System cannot reach network resources, so if any report source, output destination, or the Repository lives on a network share or UNC path, you must change the service to run as a domain account. The identity you choose here is the identity every job and every file access runs under.

1. Open `services.msc` on the server.
2. Open **Reportworq 6 Bootstrap Service** > **Properties** > **Log On**.
3. Select **This account** and enter the domain account and password. Windows automatically grants it the "Log on as a service" right.
4. Grant that account, at minimum:
    - read and write to the install folder (`C:\Program Files\Reportworq 6` by default),
    - read and write to the Repository folder if you relocate it outside the install root,
    - read and write to every network output destination,
    - read to every network report-source location the reports are pulled from.

5. Restart the Bootstrap Service so the new identity takes effect.

Warning: a path the interactive administrator can see is irrelevant if the service account cannot reach it. Report sources and destinations must be reachable by the service account. Getting the service account wrong is the most common cause of "jobs cannot read or write."

Finish setup and activate the license

1. On the server, browse to the first-run setup wizard. A fresh Windows install turns on HTTPS automatically at first boot, so use `https://localhost:8600` (the auto-generated certificate is already trusted on the server itself).
2. Work through the wizard (Welcome, Repository, License, Users, Finish). This is where you apply the license key and create the first administrator, see First-run setup.
3. To activate online, enter the license key on the License step and activate. Each activation consumes one license seat against the licensing service.

Activate an air-gapped server (manual activation)

If the server has no outbound connectivity to the licensing service, use the manual (offline) path:

1. On the License screen, choose **Manual activation** and enter the license key. Reportworq displays an **Activation Request Code**.
2. Take that request code (with the Reportworq Manual Activation form URL) to an internet-connected machine and paste it into the form. The form returns an **Activation Response Code**.
3. Paste the response code back on the Reportworq server and select **Activate**.

Tip: if IT policy blocks the in-app copy buttons, select the code text and use Ctrl+A then Ctrl+C to copy it.

Enable remote access

Local access on the server works immediately. To let administrators and users reach the server from other machines, add an inbound firewall rule for TCP 8600 (and 8601 for the Event Hub), then test with `https://<computername>:8600` (HTTPS is on by default on a fresh Windows install). Remote browsers trust the auto-generated certificate only after you distribute the trusted-root certificate to them, see Server configuration. See the network requirements in System requirements.

Uninstall

Warning: release the license before you uninstall, or the seat stays consumed against the licensing service.

1. On the **Settings > Configuration > License** screen, reveal the key (the lock icon), record it, then select **Release**. This returns the seat.
2. Uninstall from Windows **Add/Remove Programs > Uninstall**.

Warning: some files remain after uninstall. If you intend to reinstall on the same server or migrate to a new one, do not delete them, they carry the Repository and configuration state you will reuse. See Migrate to a new server.

For Linux and container deployments there is nothing to uninstall through Windows: remove the tarball directory, or stop and delete the container. The Repository lives on its mounted storage and is unaffected.

Notes and limits

- The MSI installer and the Windows service are Windows-only. Linux and container deployments use no installer and no service, see Deployment topology.
- You can change the web port later in Server configuration, which requires a restart. The default is 8600; a v5 upgrader that is not running side by side can change it back to 8300, see System requirements.

> **Going deeper.** For the license model, entitlements, and how seats are counted, see Licensing and entitlements. To stay current on releases, see Update Reportworq.

Sign-in and SSO

how-to administrator

Sign-in is the gate in front of everything: a user authenticates before any Reportworq shell mounts. Reportworq ships with a **Native** username and password provider by default, and you can instead activate a single **SSO / OIDC** provider so users authenticate with corporate identity. Exactly one provider is active at a time.

When to use it. Set the provider once as an install or policy decision. Choose Native for a self-contained account store, or an OIDC provider (Microsoft Entra ID, Google, or Custom OIDC) to let users sign in with their existing corporate identity and inherit roles from identity-provider group membership.

Before you start

- You need Administrator rights.
- The active provider is set in **Settings ▸ Integrations ▸ Authentication**. Changing it requires a service restart.
- For an OIDC provider you need its Client Id, Client Secret, and Tenant Id from the identity provider. For Microsoft Entra ID specifically, complete the Azure app registration first. See Microsoft 365 OAuth setup.

How native sign-in works

The default Native provider shows a username field, a password field, and a **Login** button. It also handles, on the same surface, these self-service flows:

- **First login** prompts a newly created account to set an initial password before first use.
- **Update Password** lets a user change their password voluntarily.
- **Password expired** forces a change at sign-in.
- **Reset password** sets a new password from an emailed reset link.
- **Forgot Password** appears only when the email distributor exists and is enabled. If email is not configured, the action reports "Missing email configuration."

Native lockout and password-complexity rules are configured on the Native provider's inline editor in **Settings ▸ Integrations ▸ Authentication**.

> **Licenses are sold as named users.** Each account is one person. Sharing a single login between several people violates the terms of service, and it also defeats the audit trail, since every action is recorded against the account that performed it. If several people need access, give each of them an account.

Switch to an OIDC provider (Entra, Google, or Custom)

1. In **Settings ▸ Integrations ▸ Authentication**, open the provider you want and enter its fields:
   - **Microsoft Entra ID**: Client Id, Client Secret, Tenant Id.
   - **Google**: Client Id, Client Secret, and optionally Tenant Id.
   - **Custom OIDC**: Client Id, Client Secret, Tenant Id, plus the **Authority URI** and each claim name (User-ID, Email, Role, Name) and one or more scopes, so it can point at an arbitrary identity provider.
2. Save. Every OIDC client secret is stored encrypted at rest.
3. Mark the provider **Active** and restart the service.

> **A restart is required.** As noted in Before you start, changing the active provider takes effect only after you restart the service. Until you restart, the previous provider stays active.

Once active, group or role claims from the identity provider resolve to Reportworq entitlements, so you can assign a user's role through identity-provider group membership instead of a local grant.

> **Switching the active provider strands accounts on the previous provider.** Changing the active provider does not simply drop permissions. The existing accounts stay bound to the previous identity provider and become reachable again only if you switch that provider back. Under the new provider, their entitlement and group grants must be re-established. Plan the cutover, because the old accounts are inaccessible while the new provider is active.

> **Before an Entra cutover, provision your way back in.** Supply an initial administrator email that already exists in Entra ID, and confirm it resolves there first. Because the switch strands the old accounts, that pre-provisioned Entra administrator is how you get back in if the cutover locks everyone out. If even that fails, use the **Recover from an administrator lockout** procedure below.

The Excel add-in login

The same native view handles the Excel and Office add-in sign-in. When the login state carries an Office cloud host, a successful sign-in posts the token back to the add-in's dialog host rather than navigating the shell, so the task pane connects without a separate browser session. HTTPS is required for the add-in.

Recover from an administrator lockout

If every Administrator is removed, or the active provider is misconfigured so no one can sign in, you can recover without a password. Recovery opens the pre-login **Administration Settings** surface, where you can repair the web-server configuration and re-establish an administrator account.

Recovery is **deliberately operator-driven**, and it cannot be gated behind a login. The person it exists for cannot authenticate; being locked out is the state it serves. So it is gated instead on proof that you control the server: either you write a file into the installation directory, or you restart the process with an environment variable set. Neither is something a browser can do.

There are two ways to open it, and both are supported:

Route	Use it when	Restart needed
`lockout-recovery.txt` in the installation directory	A normal Windows or Linux installation. This is the easier route on a Windows service.	None, in either direction
`RW_CONFIG_ACCESS=1` in the environment	The installation directory is read-only or is replaced on every restart, such as a Docker or Azure App Service container. See Deployment topology.	Once to open, once to close

Open recovery with a file

1. **Take the path from the log.** Every startup writes an Information line naming the **full path** of the file to create, whether or not the file exists. That line is there for exactly this moment: you are locked out of the screen that would have told you anything, and the installation directory is the one thing you cannot work out from the outside.
2. **Create a file named `lockout-recovery.txt` at that path.** Only the presence of the file matters. Its contents are never read, so an empty file is enough.
3. **Refresh the sign-in page.** The logon screen now shows an **Administration Settings** cog. You can reach it from anywhere, not only from the server console.
4. **Open it and recover.** You can fix the web-server configuration (name, port, SSL) and reach Security to re-establish an administrator account or repair the active provider. Only system-scoped tabs exist in this state, because no workspace provider is loaded.
5. **Delete `lockout-recovery.txt`.** The cog disappears and recovery closes.

Neither opening nor closing needs a restart. The file is checked as the sign-in screen renders, so creating it opens recovery and deleting it closes recovery within a couple of seconds. That is the point of this route: on Windows, Reportworq normally runs as a service, so setting a process environment variable means editing the service definition and restarting the service twice, once to open recovery and once to close it again, at exactly the moment someone is locked out of their own instance.

> **The installation directory is the folder holding the running binaries, and on Windows that is not the install root.** The Bootstrap Service starts the application from a versioned subfolder, so the path is of the form `C:\Program Files\Reportworq 6\Versions\v6.0.0.0`, not `C:\Program Files\Reportworq 6`, and it changes when you update. Take it from the log line rather than typing it from memory.

> **Nowhere else works.** Only that one folder is checked. A copy of the file at the drive root, in the Repository, or in any other folder does nothing at all, and each of those locations was considered and rejected on security grounds. If you created the file and no cog appeared, the likeliest cause is that it is not in the folder the log named.

Open recovery with an environment variable

Use this route where the installation directory is read-only or ephemeral, such as a container image.

1. **Restart Reportworq with** `RW_CONFIG_ACCESS=1` set in its environment.
2. **Browse to the instance.** As with the file route, you can do this from anywhere. The logon screen shows the **Administration Settings** cog.
3. **Open it and recover**, exactly as in the file route above.
4. **Restart again without the variable.** This gate stays open until the process restarts without it.

While recovery is open

- **The pre-login administration shell is reachable by anyone who can reach the instance**, not only by someone at the server console. Recovery is a whole-instance switch, not a local one.
- **Nothing closes it for you.** The file is never deleted automatically and the variable never expires, because a recovery that timed itself out would strand the administrator it exists for. Closing it is a deliberate step: delete the file, or restart without the variable.
- **The instance says so in its log.** Reportworq logs a warning when recovery opens, repeats a standing reminder while it stays open, and logs a line when it closes. An instance somebody opened for a five-minute repair and forgot to close is visible in the log rather than silent.

Recover, then close it. This is also why keeping at least one Administrator, and provisioning an Entra way-back-in before a cutover, both matter: either route needs someone with access to the server itself, to write a file into the installation directory or to restart the service.

Notes and limits

- Only one provider is active at a time.
- Sign-in itself grants nothing. Role and per-item access are governed after sign-in by workspace security. See Roles and what you can do.
- A user with no workspace access cannot enter a shell. Membership of at least one workspace is required to log in. See Workspaces.
- Native is the only provider with self-service password flows.

> **Going deeper.** To release a license seat immediately after removing a claims-based role, clear the account's "Last Claims." See Accounts, groups, and entitlements.

Distribute the load

how-to administrator

This page is about **provisioning and placement**: deciding how much hardware a workload needs, and making sure the work actually spreads across it. It answers the question customers ask most often, which is some version of "how many servers do we need, and why did adding one not help as much as we expected".

When to use it. Before a new deployment, before a close window you expect to be tight, when adding nodes to an existing cluster, and whenever throughput is not scaling the way the server count suggests it should.

Before you start

- You need the **Administrator** entitlement. Every setting on this page is under **Settings**, which is administrator-only, and the **Performance** page is not offered on a load-balancer node.
- Read Performance and capacity planning first if you have not. In particular, distribution and contribution are sized separately, and this page is mostly about distribution.
- Have the baseline requirements to hand: System requirements.
- Scaling out to more than one server needs a **Load Balancer** node count of 1 or more on your license. See Licensing and entitlements.

Size the server before you size the settings

Two settings on **Settings ▸ Configuration ▸ Performance**, under **Execution**, set how much a server takes on at once.

Setting	Default	Range	What it limits
Parallel Job Execution	4	1 to 50	Server-level concurrency: how many top-level jobs run at once
Tasks per Job	3	1 to 50	Per-job concurrency: how many outputs or variations run at once **inside** one job

> **Two different controls share the name "Parallel Job Execution".** On **Settings ▸ Configuration ▸ Performance ▸ Execution** it is the server's own top-level job concurrency, and it accepts **1 to 50**. On a load-balancer node's own settings screen, and as the **max-jobs** spinner on the main server's Load Balancer grid, it is that node's routing capacity, and it accepts **0 to 128**. They are stored separately and do different work. Check which screen you are on before you change one.

Their product is the theoretical maximum number of concurrent tasks. At the defaults that is 12; at 5 by 5 it is 25.

The rule that governs the whole page: peak memory scales with the number of jobs in flight. Allow roughly 2 GB per concurrent job on top of the base footprint, so five parallel jobs want about 10 GB free at the peak. Size these settings against the RAM the server actually has, not against its core count. A server tuned past its memory does not degrade gently: it runs out of memory under load.

Size against your **peak** concurrency rather than your average. The busiest ten minutes of the close is what the machine has to hold.

On Azure, the same arithmetic picks the plan. A **B2** plan is enough for a proof of concept or a pilot; production wants **P1v3 or above** for the RAM and throughput a real job load needs. Run exactly one web instance, which owns the UI and the scheduler, and add worker nodes against the same shared repository for throughput. See Deployment topology.

What a cluster distributes, and what it does not

When one server cannot finish the load inside its window, scale out into a **load-balanced cluster**. Getting value from it depends on knowing exactly what it spreads.

It distributes whole jobs, and a job is atomic. Each job runs start to finish on a **single node**. A node that is not allowed to run a queued job re-queues it so another node picks it up. This is how batch job execution, Excel reporting, and campaign form load spread across the cluster.

It does not distribute interactive Excel add-in refresh. When an analyst refreshes a Data Model view or a Direct Query in the add-in task pane, that query runs on whichever node holds their session, whatever the load balancer's capacity settings say. Distributing interactive refresh is planned for a later phase. Today, plan interactive analyst load against the node people sign in to, not against the cluster total.

Contribution session placement has its own rule. When a contributor opens a form, Reportworq checks every online server and steers the new session to the one with the most available memory that can still accept a session. That is a separate mechanism from job routing, and it is governed by the 90% memory admission valve rather than by node capacity settings. See Scale and size a contribution campaign.

Design jobs so the cluster can spread them

This is the answer to "we added a node and throughput barely moved", and it is a job-authoring decision rather than a settings one.

Because **a job runs atomically on one node**, a single job can never use more than one node's capacity, no matter how many nodes you own. The two levers are therefore different in kind:

- **Split the work into more jobs** to use **more nodes**. Ten jobs can occupy ten nodes. One job cannot.
- **Raise Tasks per Job** to use **more of one node**. Within a single job, its outputs and variations run in parallel up to that limit, on the node that owns the job.

So, when a batch is not spreading:

1. **Count the jobs, not the outputs.** A close batch expressed as one enormous job pins itself to one machine. The same work expressed as one job per region, entity, or pack is what lets the cluster work.
2. **Check Tasks per Job against the shape of each job.** A bursting job producing forty outputs on a node whose Tasks per Job is 3 is running three at a time on purpose.
3. **Remember both limits are per server.** Each server runs up to its own Parallel Job Execution and its own Tasks per Job, so cluster capacity is the per-server figure multiplied by the server count. Grow throughput by adding nodes rather than by pushing one server's limits past what its RAM can hold.

One more shape worth knowing. An instance with several workspaces shares **one job queue** across all of them. Rows belonging to other workspaces are redacted and cannot be expanded or canceled until you switch to that workspace. So one workspace's batch is not isolated from another's by the workspace boundary, and a heavy tenant can occupy queue capacity that another workspace is waiting on. Plan the clock across the whole instance, not per workspace.

Set per-node capacity

A load-balancer node presents a reduced shell at its own address, with an **Execution** category carrying:

- A **Server Instance Name**, which identifies it in job screens, the Load Balancer monitor, and the node pivot on **Settings ▸ Performance ▸ Distribution**. Set something meaningful here: it is the label every later diagnosis reads.
- An **Enable Execution** checkbox, which is what decides whether the node takes work at all.
- A **Parallel Job Execution** value from **0 to 128**, which caps how many jobs route to that node.

These are the same three values as the node's row on the main server's **Settings ▸ Load Balancer** grid, where they appear as the node name, the **enable** switch, and the **max-jobs** capacity spinner. Edit them from either end.

Size capacity against the machine, not against the cluster. A node's max-jobs value should not exceed what its RAM can actually hold, because peak memory scales with jobs in flight and the routing limit knows nothing about how much memory the machine has.

Concentrating execution on dedicated nodes. A node's **Parallel Job Execution** accepts **0**, and its **Enable Execution** checkbox can be cleared, so a node can stay registered in the cluster and take no work. That is how you drain a node ahead of maintenance without removing it and having it re-register.

The main server's own **Parallel Job Execution**, on Settings ▸ Configuration ▸ Performance, is the other control described above and accepts **1 to 50**; it cannot be set to 0. So the main server always keeps some execution capacity of its own. To concentrate batch work on dedicated nodes, lower the main server's value and raise the nodes', rather than trying to zero the main server. Given that interactive add-in refresh is not distributed and lands on the node holding the session, leaving the main server a small amount of capacity while the nodes carry the batch is usually the split you want anyway.

Taking a node out of service. Disable it, or lower its max-jobs, before patching its host, so no new jobs route to it. **Remove** deletes the entry but a still-running server re-registers itself and reappears; stop its service to retire it for good.

Pin work to specific nodes with Filter jobs

The per-node **Filter jobs** action opens a tri-state tree (include, exclude, or inherit) over the instance's workspaces, folders, and jobs, with two modes: **Include all jobs and workspaces**, which is the default, or **Include or exclude specified jobs and workspaces**.

Reportworq saves the filter on the node and enforces it when routing. For each queued job it checks the **job**, then walks up its **parent folders**, then its **workspace**, and re-queues any job the node is not allowed to run so another node picks it up.

Filters are a placement tool, not a throttle. Three uses earn their keep:

- **Keep a long job off the fast lane.** Exclude a single multi-hour job from the nodes that serve short interactive-feeling runs, so it cannot occupy their slots.
- **Match a job to a node that can reach its data.** A job whose source workbooks live on a network folder only some nodes can reach belongs on those nodes. This is a correctness problem before it is a performance one, and the filter is where you express it.
- **Reserve a node for one workspace or one folder.** Useful where one team's close cannot be allowed to queue behind another's.

The trap. Every exclusion narrows the pool of nodes eligible for that job. Exclude a job everywhere and it never runs. Filters are worth auditing after node changes, because a filter written against a three-node cluster can quietly strand work when a node is removed.

Spread the clock

Capacity is not only how much hardware you own; it is how much of it is idle at 07:59 and oversubscribed at 08:00. Two facts make this concrete.

The 30-minute overdue rule. When the scheduler cannot run an occurrence at its due time, because the master scheduler was paused, the host was down, **or a prior run of that schedule was still going**, the occurrence becomes overdue. An occurrence more than **30 minutes** overdue is **skipped, not run late**. This is a fixed rule and not a setting.

That last cause is the capacity-planning one. A schedule whose runs regularly take longer than its interval will start missing occurrences, and past 30 minutes they are silently skipped rather than queued. The symptom looks like missing output rather than slow output, which is why it is worth checking the schedule shape before checking the server. To force a skipped run, use **Run now** on that schedule's row.

Everything at the top of the hour is a spike everywhere. A single clock time concentrates job execution, source system load, and outbound mail into one moment. Stagger schedules across the window you actually have rather than across the minute you happen to have picked, and give the heaviest jobs the start of the window rather than the end of it.

Schedule health, the overdue footer, and the failure counts are on **Scheduled Content**. See Manage scheduled content.

Keep the tiers out of each other's way

Three isolation settings decide whether one kind of work can hurt another. All three ship on, and the useful thing is knowing what each protects.

Setting	Where	What it protects
Enable Out of Process Job Execution	Settings ▸ Configuration ▸ Performance	Runs job execution outside the web app process and frees its memory after each run, so a heavy job cannot pressure the web tier. Leave on unless support advises otherwise.
Enable Out of Process Reporting Queries	Settings ▸ Configuration ▸ Performance ▸ Reporting	Runs an analyst's interactive add-in query in a dedicated worker rather than inline on their web session, so a runaway query cannot stall other users or exhaust the web app.
Enable Out of Process Contribution Sessions	Settings ▸ Configuration ▸ Performance ▸ Contribution	Runs form calculation in separate worker processes, which is what makes the per-server memory console and the admission valve possible.

Two behaviors of the reporting worker are worth knowing before you rely on it:

- **It is retired when idle and respawns on demand.** A worker that has been idle past its threshold is shut down so its warm connections and caches are released; the next query for that workspace transparently starts a fresh one. A cold start is therefore normal after a quiet period rather than a fault.
- **If the worker cannot start, Reportworq falls back to running the query in the web app.** The failure is written to the host log and the fallback holds for **five minutes** before another attempt, so a host problem does not leave analysts staring at a spinner. The toggle continues to read as enabled while this is happening, so the **host log is where you confirm it**, not the settings screen. A clean start clears the fallback immediately, so a fixed host goes straight back out of process.

The **Reporting** category carries three more settings beside the toggle.

Setting	Default	What it does
Reporting Log Level	Warning	The log level passed to the reporting worker.
Worker Start Timeout	90 seconds	How long to wait for a spawned worker to load its plugins and connect before failing the query. The worker hosts the full connector stack, so its cold start is heavier than a contribution worker's.
Query Timeout	300 seconds	How long to wait for a single query to return. This is a ceiling on how long one query may occupy a worker, so raise it only for a genuinely long query you intend to support.

The idle threshold that retires a quiet worker is **30 minutes** and has **no control on this screen**; it is a settings-file value.

Housekeeping that protects capacity

These do not make a job faster, but they stop an instance getting slower over months.

Lever	Where	Why it matters
Default History Retention (30 days)	Settings ▸ Configuration ▸ Performance ▸ Execution	Bounds how much run history and stored output accumulates. Per-folder and per-report policies, and pinning, are on Output retention.
Compress History Files (on)	Settings ▸ Configuration ▸ Performance ▸ Execution	Compresses job-history databases at rest. Performance measurements honor it too.
Log Retention (90 days) and **Audit retention** (90 days)	Settings ▸ Configuration ▸ Logging	Logs and audit files grow with activity. A value of 0 on audit retention keeps them indefinitely, which is a deliberate choice rather than a default to leave unexamined.
Enable Debug Logging (off)	Settings ▸ Configuration ▸ Logging	A live instance-wide switch that produces very large logs and HTTP-Archive files. It sits outside the save flow and takes effect immediately. Turn it on only when support asks, and off again after the capture. Leaving it on is a real and avoidable drag.
Recycle Service (off)	Settings ▸ Configuration ▸ Web Server	An automatic daily restart at a chosen time to free resources. Users refresh their browsers afterwards, so pick an hour nobody is working.
Clear Repository Cache	Settings ▸ Configuration ▸ Performance, under **Advanced Features**	An advanced action in a category that is collapsed by default. Use it when support asks rather than as routine maintenance.

Retention is also what bounds your diagnostic history. Performance measurements are stored with the run and go when the run goes, so a 30-day retention gives you a 30-day comparison window. If you compare period-close runs year on year, pin them.

A provisioning sequence that works

1. **Estimate from the baseline.** Start at 32 GB plus roughly 2 GB per concurrent job, and decide the peak concurrency the window requires. See System requirements.
2. **Set concurrency to what the RAM supports**, not to the core count, using Parallel Job Execution and Tasks per Job.
3. **Run a representative batch and measure it.** The **Performance** tab reports each run's peak memory and maximum CPU, and **Settings ▸ Performance ▸**

Distribution reports the median run across the estate. That is your real per-job cost, replacing the 2 GB rule of thumb.

4. **Check the shape of the work before adding hardware.** If throughput is not scaling with nodes, count the jobs, because a monolithic job cannot use a second machine.
5. **Add nodes, then size each one.** Open ports 8600 and 8601 on every server, put the Repository on a UNC share every node can reach, set **Web App Host** on the main server before starting any node, and run the same version everywhere. See Deployment topology.
6. **Place the work.** Set per-node max-jobs and Parallel Job Execution, consider the coordinator pattern, and use Filter jobs where a job belongs on particular nodes.
7. **Spread the clock**, and confirm no schedule is regularly overrunning its own interval.
8. **Size contribution separately** if the deployment collects input. See Scale and size a contribution campaign.
9. **Re-measure after the first real close** and adjust. The first estimate is an estimate; the second one is evidence.

Notes and limits

- **Concurrency limits are per node.** Cluster capacity is the per-node figure multiplied by the node count, and there is no instance-wide concurrency ceiling to set in one place.
- **A job never spans nodes.** No setting changes this. Job granularity is the lever.
- **Interactive add-in refresh is not distributed across the cluster** and lands on the node holding the analyst's session.
- **Load balancing is licensed** as a node count; 0 disables it, and a single-node install shows an empty node grid.
- **A node's Status reflects a successful job-dequeue attempt in the last two minutes**, so it reports participation rather than health.
- **The 30-minute overdue window is fixed** and cannot be configured.
- **There is no throttling by priority.** Reportworq has no job priority or queue-weighting control. Placement is expressed through node capacity, filters, and the clock.

> **Going deeper.** For diagnosing content that is already slow, see Manage and improve performance. For the cluster mechanics, node join, and cloud shapes, see Deployment topology. For every setting named here, see Server configuration. For contribution sizing arithmetic, see Scale and size a contribution campaign.

Log diagnostics and support packets

how-to administrator

Settings ▸ Log Diagnostics builds a support packet, a single zip of selected logs and diagnostic data, to send to the Reportworq support desk. It replaces the older blunt whole-folder send and clear actions with targeted selection, multi-node reach across a cluster, and a resilient build-then-upload-or-download flow. This guide is for the administrator preparing a support case.

When to use it. When troubleshooting a defect or performance issue with the support desk, especially across a load-balanced cluster. For compliance records instead, use Audit logs.

Before you start

- You need administrator (author or system-admin) rights.
- Uploading needs connectivity to the support desk. Download works offline.
- Capture the right detail first, see the logging rules below, because you cannot add detail to a run that already happened.

Set logging before the run you want to diagnose

The single most important rule: turn on the detail you need before you re-run the failing job, not after.

- **Enable Verbose logging before the run.** Per-run verbosity is set in the Run Job Options screen. Verbose is the normal diagnostic level. Turn it on, then re-run the failing job, so the detail is captured into that run's archive.
- **Avoid Trace unless support asks.** Trace is far more detailed and carries a real performance impact. Enable it only when support explicitly requests it, and turn it off afterward.
- **Turn Debug Logging off after capture.** The instance-wide Debug Logging switch, under Configuration ▸ Logging, emits HAR and HTTP-trace files that grow fast and can fill the disk. Enable it only on request, and disable it once you have the capture.

Choose what goes in the packet

Open **Settings ▸ Log Diagnostics**. A toggle at the top switches between two selection modes.

Simplified mode (the common case)

1. Leave the mode on **Simplified** (the default).
2. Turn on **include log files in a date range** and set the From and To pickers. A live hint shows how many log files fall in the range.
3. Optionally tick the specific job runs to include; each ticked run adds its execution-database zip.
4. If you already have an open support case, enter its **ticket id**, and add a short **description** of the problem. Both are optional and are written to `support-info.txt` at the packet root. You do not create a case here; the ticket id only links this packet to a case that already exists.

Advanced mode (surgical selection)

1. Switch the mode to **Advanced**.
2. On the **Logs** tab, tick individual files or whole folders from the install Logs folder (loose app logs, the Http folder of HAR and HTTP-trace files, and the ContributionSession folder).
3. On the **Load balancer** tab, expand each cluster node and tick the logs to pull from it over the network. A single-node install sees "No load-balancer nodes are configured." An offline or slow node is omitted and the rest still build.
4. On the **Jobs** tab, tick the job-history runs to include; each adds its execution-database zip.

Switching modes invalidates an already-built packet, because the two modes select different things.

Optional: include the repository data folder

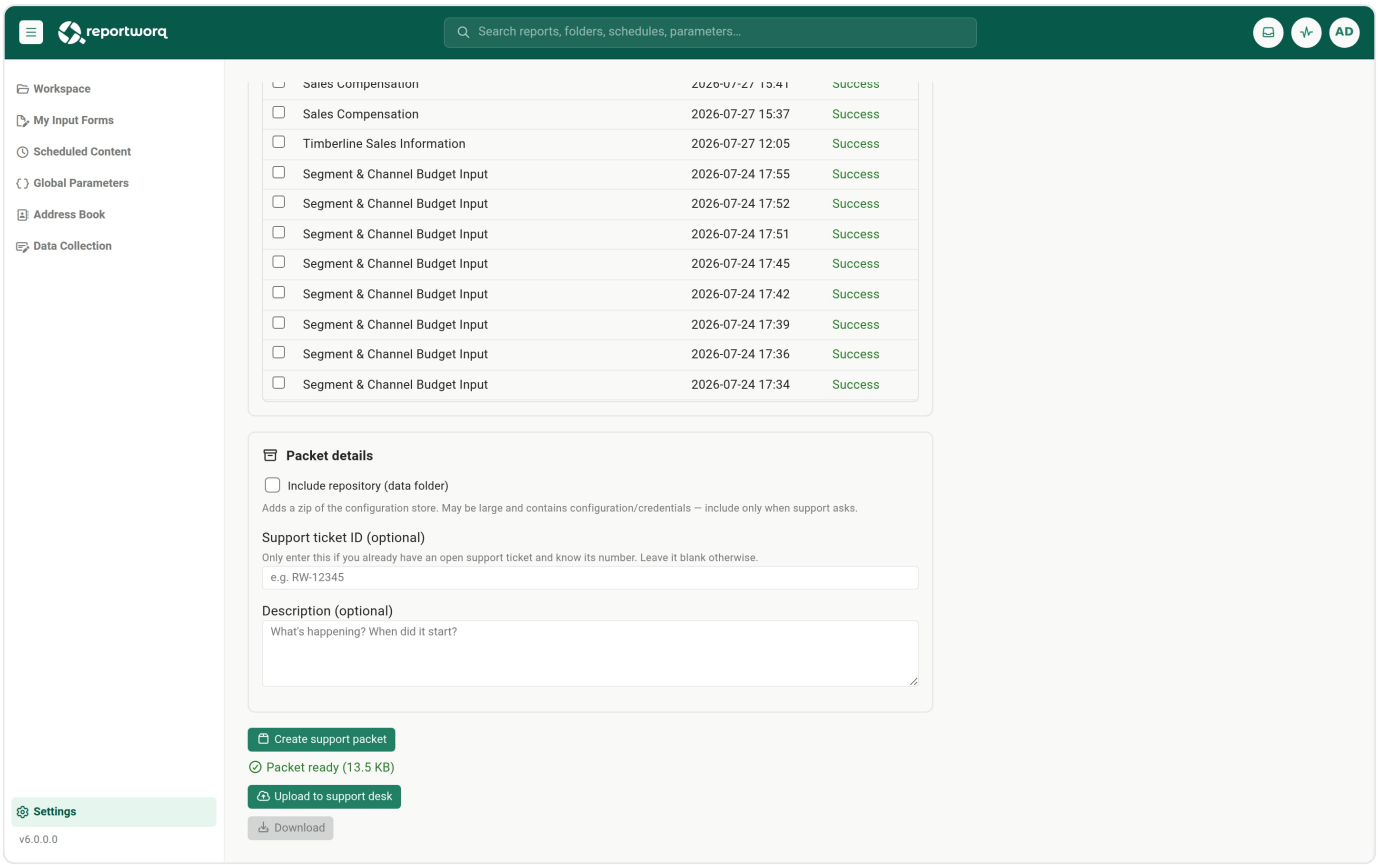
Both modes share a toggle, **Include repository (data folder)**, that adds a zip of the repository **data** folder (the raw config-store data). It is off by default. It can be large and contains configuration and credentials, so include it only when support asks.

This is the raw config-store data zip. It is **not** the Repository metadata report, which is a separate, human-readable Excel export of the instance's jobs, reports, and schedules.

Create the packet, then upload or download

1. Select **Create support packet**. This is enabled once you have at least one selection. Assembling and caching the zip takes a moment while Reportworq gathers the selected files.
2. When the packet is ready, **Upload to support desk** and **Download** buttons appear.

3. Select **Upload to support desk** to send it, or **Download** to save the same prepared zip locally.
4. If an upload fails on a large packet or a slow link, your selections and the built zip are kept, so you can simply **Download** and send it another way without rebuilding. Changing any selection clears the packet, so create a fresh one.



Send diagnostics from a single run

For one failing execution, you do not need the whole packet. Open the run in job history, then **Diagnostics ▸ Send to Support**, to upload that execution's logs securely. This is the per-run counterpart to the Log Diagnostics screen. See Diagnose job failures.

Housekeeping

- **Delete** removes a selected local file or folder.
- **Clear all logs (except active)** deletes every local log except the current active (locked) one, which is protected and keeps recording. Use it to free disk after an incident.

What is in the packet

The zip is laid out as:

- `support-info.txt` at the root: ticket id, description, generated time, instance id, version, and contents.
- `local/…`: the selected local logs.
- `remote/{nodeName}/…`: per-node logs collected across a cluster.
- `jobs/{jobName}_{execId}.zip`: the selected job execution databases.
- `repository/data.zip`: only if repository metadata was included.

Support intake checklist

When you open a case, include the product version (shown in the app footer), a symptom description, the exact error text, the steps to reproduce, and any sample data or workbooks. Reach support at support@reportworq.com, at forum.reportworq.com, or through the "Create A Support Ticket" web form.

Notes and limits

- This is a support-packaging tool, not the audit viewer. For compliance and change-tracking records, use Audit logs.
- The repository-metadata option contains credentials; include it only on request.
- Multi-node collection only does anything on a real cluster; a single-node install sees an empty Load Balancer tab.

> **Going deeper.** For reading and AI-analyzing a single run's failure, see Diagnose job failures. For symptom-to-fix pairs, see Troubleshooting index.

First-run setup

how-to administrator

On a brand-new install, Reportworq boots into a one-time, full-screen setup wizard instead of the sign-in screen. The wizard gets the server to a usable state: it chooses where the Repository stores its data, applies the license, and creates the first administrator account. It runs once. After you finish it, all further administration happens in **Settings**, and you can reopen the wizard from there if you need to reconfigure the Repository later.

Before you begin

- A completed install with the Bootstrap Service running, see Install Reportworq.
- Your license key.
- The credentials you want for the first administrator account.

Open the wizard

1. On the server, browse to `https://localhost:8600`. On a fresh install with an empty Repository, this opens the setup wizard rather than the sign-in screen. (A fresh Windows install serves over HTTPS by default, see System requirements.)

Work through the steps

The wizard advances through five named steps:

1. **Welcome.** The intro step ("Get up and running in a few steps!"). Continue.
2. **Repository.** Choose where Reportworq stores its data, and how this install obtains its Repository. Pick one of four options (see below).
3. **License.** Enter your license key to apply the license. For a server without connectivity to the licensing service, use the manual activation path described in Install Reportworq.
4. **Users.** Create the first Reportworq administrator account. This is the identity you will sign in with after setup.
5. **Finish.** Setup completes and reloads into the normal app, which presents the sign-in screen.

Repository options

The Repository step offers four choices:

- **Create a new repository.** Start fresh at an empty or unused path. This is the usual choice for a new install, and it leads on through the License and Users steps.
- **Import a Reportworq v5 repository.** Create a new v6 repository from a copy of an existing v5 repository. Use this to upgrade a v5 instance; the v5 repository is copied and left unchanged.
- **Connect to an existing repository.** Point this install at a repository that already contains Reportworq **v6** data. **Do not point it at a v5 repository.** To bring a v5 instance forward, use **Import a Reportworq v5 repository** above, which creates a new v6 repository from a copy and leaves the v5 original untouched.
- **Add a load balancer to an existing web app.** Run this install as an additional execution node that shares the primary web app's repository, see Deployment topology.

The three existing-repository choices (import, connect, and load balancer) apply the choice and restart the service, then present a short sign-in path. Only **Create a new repository** continues on to the License and Users steps in the same session.

When you complete a new-Repository install, Reportworq seeds a workspace named **Default** so you land in a usable workspace after signing in.

The security defaults, and when to tighten them

A fresh install deliberately starts **open**, so that the person who just finished the wizard can actually use it. Two defaults do that, and both are worth understanding on day one:

- The **Administrator** entitlement is seeded with **All Users**, so every enabled account is an administrator.
- The **Default** workspace's member entitlement and its **root folder ACL** are seeded with **All Users**, so every account can reach everything in that workspace.

This is a bootstrap posture, not a recommendation. **If more than one person will use this environment, tighten it before you add accounts:**

1. Remove **All Users** from the **Administrator** entitlement and name the specific accounts or groups that should administer the environment. This matters most under single sign-on, where accounts are provisioned automatically at first login: leave it as seeded and everyone who signs in becomes an administrator.
2. Decide what the workspace root should be. Leaving **All Users** on the workspace root folder is **perfectly reasonable if you are not using folder-level security** and everyone in the workspace is meant to see everything in it. If you do intend to secure by folder, replace it with the groups that should have access, because a permissive root undercuts the folder ACLs beneath it.

See Accounts, groups, and entitlements for the entitlement catalog and how membership resolves.

After the wizard

1. Sign in with the administrator account you just created.
2. You land in the **Default** workspace, ready to configure connections, providers, and distributors.

Notes and limits

- The wizard runs only on an empty Repository. A server whose Repository is already configured, or that already has accounts, does not show the full wizard again on its own.
- If a new-Repository install needs a restart to bind the Repository, the wizard resumes at the **License** step rather than starting over.
- A node pointed at an existing Repository (for example a load-balancer node) takes a shorter "Restart and login" path instead of the full new-Repository finish, see Deployment topology.
- You can reopen the wizard later to reconfigure. To change the Repository path, connect or import a repository, or convert this install into a load-balancer node after setup, go to **Settings > Configuration > Web Server** and select **Reconfigure**. Manage accounts under Security, see Accounts, groups, and entitlements.

> **Going deeper.** For the license model and entitlements applied in the License step, see Licensing and entitlements.

Manage and improve performance

how-to administrator report author

Distribute the load is about the hardware and where the work runs. This page is about the **work itself**: why an existing report, job, or form performs the way it does, and what to change.

When to use it. When a pack that used to finish in three minutes now takes twenty, when several people report slowness in the same week, when everything feels slower after an upgrade, or when you have been asked to make something faster and want to know where the time is actually going before you start changing things.

Before you start

- Open a report's **Run History** from the workspace browser: the report row's **...** menu, then **Run history**.
- The diagnostics tabs sit behind **View diagnostics**, which a view-only **User** account does not have. Authors and administrators reach them.
- **Settings ▸ Performance** is administrator-only and is not offered on a load-balancer node.
- Comparison needs history. A baseline needs at least **3** comparable earlier runs; a step-change search needs **6**. A recently upgraded instance is thin until new runs accumulate.

Ask the questions in this order

Four surfaces answer four different questions, and working from the outside in stops you optimizing the wrong thing. A report that regressed at the same moment as every other report on its node is not a report problem.

Ask	Where	You learn
1. **Is it the instance?**	**Settings ▸ Performance ▸ Distribution**	Whether many reports moved together, and whether a **release** or a **node** explains it
2. **Is it this report, over time?**	The **Compare** tab on a run's diagnostics	Whether there is a single point where this report's typical run time stepped, and whether volume, release, or node moved with it
3. **Where did this run's time go?**	The **Performance** tab on a run's diagnostics	The cost broken down by output, workbook, processing phase, and calculation engine
4. **What did it coincide with?**	The **Timeline** tab on a run's diagnostics	Every span, event, and the memory and CPU trace, drawn on one clock

If contributors rather than reports are the problem, the equivalent funnel is per-server memory on **Settings ▸ Performance ▸ Contribution**, then a form's footprint in **Campaign profiling**. See Scale and size a contribution campaign.

1. Rule the instance in or out

Settings ▸ Performance ▸ Distribution ranks every report against its own baseline and pivots the whole estate by product release and by execution node. It is the only surface that can see across reports, and it is where "did the upgrade do this" is answered.

Choose a window of 30 days, 90 days, or 1 year. Everything on the page, including the baselines, is recomputed inside that window, so a delta on screen is always explained by runs you can see.

Read the five figures first. **Runs with telemetry** tells you how much of the window is actually measurable, and a low percentage means the page below is thinner than it looks. **Releases seen** and **Nodes seen** tell you whether the pivots have anything to compare at all: one release or one node is nothing to compare.

Then read the two pivots at the foot of the page. Three rules govern them, and they are the reason a cell sometimes refuses to give you a verdict:

- **A group is compared against the other groups, never against the overall median.** A node that runs most of the work would drag an overall median toward itself and look normal however slow it was.
- **A group needs at least 3 runs**, and there has to be something to compare it with.

- **Volume has to match.** If a group processed more than 15 percent more or less data than the others, the cell reads "different volume" and gives no verdict. A release that looks 40 percent slower because its runs carried three times the data is not a regression.

A refusal to compare is information. It is the page declining to tell you something it cannot honestly know, and it is far more useful than a confident number built on three unlike runs.

The page distinguishes its two empty outcomes, which matters when you are diagnosing rather than browsing. **"No runs in this window."** means it read the history and found nothing, so widen the window or check that job-history logging is on. **"Performance data is unavailable."** means the read itself failed, and it shows what went wrong. Those are different problems and the page will not pass one off as the other.

Full field-by-field detail is in Compare performance across reports.

2. Find the moment a report changed

The **Elapsed vs baseline** column answers "was the last run normal". That is the wrong question for a report that has been drifting for a fortnight, because every individual run looked fine against a baseline that was itself creeping upward.

The **Compare** tab asks the other question: across this report's whole history, is there a point where its typical run time changed? It searches for a single step and either finds one or says plainly that there is not one.

The verdict sentence is the whole answer, and it names a cause only when the data shows one. It will tell you whether **volume** moved across the same split, whether the **release** differs on either side, and whether the runs moved to a different **node**. "It got slower" and "it got bigger" are different findings, and this is the surface that separates them.

On a bursting job, use the scope picker to narrow to **one variation**. A whole-run total moves for reasons that have nothing to do with any one output: a burst list grew, a region was retired, a run was restricted to one recipient to test a fix. Each surviving output stays perfectly comparable with its own past even when the total does not.

See Find when a report got slower.

3. Break one run down

The **Performance** tab leads with one sentence naming the dominant cost, then six figures, then a walk you can drill.

Read Elapsed with its volume denominator. Underneath the elapsed time sits the volume the run covered: data points where the run recorded them, otherwise the formula count. This is the first thing to check, because a report doing twice the work in twice the time has not regressed.

Walk from the run down to a phase. One ranked list, largest first, at three levels:

1. **Whole run**, listing the outputs it generated. This is where a burst tells you which output is expensive.
2. **One output**, listing the workbooks inside it.
3. **One workbook**, listing its processing phases. Phases are the last level, and this is where "the PDF took nine minutes" or "the refresh took nine minutes" gets settled.

At workbook level, two tails are listed apart from the ranked phases: phases that **ran but cost nothing measurable**, and phases that **never ran at all**. Keeping them separate matters, because "this workbook has no charts" and "charts were free" are different facts about the same run.

What each phase is

The phase names are the vocabulary the whole surface is written in, so knowing what each one covers is what turns the ranked list into an action. These are the phases Reportworq expects a workbook to have, in roughly the order they run.

Phase	What it covers
Open Workbook	Loading the source workbook from its report provider, file open and parse
Workbook Initialization	Preparation before calculation: hidden-sheet extraction, very-hidden sheet removal, worksheet generation
Generate Worksheets	Cloning template sheets per parameter set
Data Refresh	The full calculation pass: every datasource calculation service plus the recalculation loop
Recalculate Indirect Formulas	`INDIRECT` and dynamic-formula recalculation after structural edits
Suppress Worksheets	Sheet-level suppression driven by `RWSHEETSUPPRESS`
Rename Worksheets	Sheet renaming driven by `RWSHEETNAME`
Variables	Report-scoped variable resolution
Pivot Tables	Pivot-table filter application
Charts	Refreshing add-in-authored chart and map pictures against the calculated workbook
Images	Shape and image refresh across the workbook
Formatting	Tab colors, output order, and exception-cell reads
Remove Formulas	Formula stripping before save
Workbook Cleanup	Deleting worksheets that are not part of the output
Save Workbook	Serializing the destination workbook to its output stream
Save PowerPoint Datasource	Saving the extra workbook copy kept as a PowerPoint datasource
Script Hooks	Custom scripts that run inside report processing

Data Refresh is almost always the largest, and it is the one that contains the recalculation loop, which is why the engine table below it matters more than the phase list. When something *other* than Data Refresh tops the list, that is the interesting result: a workbook dominated by **Charts**, **Images**, or **Formatting** is paying for its presentation rather than its data, and that is a different conversation with the author.

A few other phases can appear when the work applies to that report, such as parameter application, closing the workbook, or loading a non-Excel source document. They are ranked like any other; only the phases in the table above take part in the "never ran" tail.

The engine table is where the fixable problems are

Below the walk, the **Engines** table lists one row per formula family and per calculation service, with **Calls**, **Total**, **Avg**, and **Recalc passes**.

Recalc passes is the column to read. It says `once` for an engine that ran a single time, or `N of M` for one that re-ran during recalculation. An engine reading `5 of 5` on a five-pass workbook ran on **every** pass, and Reportworq highlights that row when the pass count is high enough for it to be expensive.

That pattern is the single most actionable finding on the whole surface. A data retrieval repeated on every recalculation pass is usually a formula that can be restructured, not an unavoidable cost, and it is invisible from the durations alone: an engine can look cheap per call and still dominate a run because it is trapped in the loop.

Above workbook level the pass counts are folded together from the workbooks in scope, so `5 of 5` there reads as "in the recalculation loop the whole way" rather than as a run-wide total.

See Review a run's performance.

4. See what the time coincided with

The **Timeline** tab re-projects the same run onto one clock: execution phases with their varieties nested beneath, per-workbook phase spans, event ticks, and the memory and CPU trace, all sharing one x-axis and one cursor.

It answers a different question from the other tabs. They say where the time **went**; this says what the time **coincided with**, which is the only way to see that the third data refresh landed on a memory spike.

Two readings are easy to get wrong:

- **A hatched break in the CPU line is "not measured", never 0 percent.** The opening stretch of every run is hatched, because the first sample has no interval to measure against. That is correct, not missing data.
- **The expected-completion band is the expectation that stood when the run started**, built from the runs before it, which is what makes an estimate on a finished run honest. It can legitimately extend past the end of the run, since an estimate built from a median is beaten by about half of all runs.

The tab also says when a cap has hidden detail, such as a very long run whose resource samples were thinned or a run with more spans than the chart draws. Read those notes before concluding that something did not happen.

The levers, and the order to try them

Work top down. Everything above a line is cheaper and more reversible than everything below it.

Content: change what the work does

Lever	When it applies	What to do
A formula family in the recalculation loop	The engine table shows `N of N` on a multi-pass workbook, or a highlighted row	Restructure the formula so the retrieval is not re-evaluated on every pass. This is usually the largest single win available.
Data volume grew	Elapsed rose and the volume denominator rose with it	Not a regression. Decide whether the report needs that volume, and narrow the query or the burst if it does not.
One output dominates a burst	The whole-run list is lopsided	Treat that output as its own problem. The other outputs are fine and changing them costs effort for nothing.
A phase you did not expect	The workbook-level list ranks something surprisingly high	The phase names what kind of work it was. Confirm the report actually needs it.

What the breakdown will not tell you. It attributes cost to phases and engines, not to individual design choices, so it cannot confirm that a particular chart, image, or block of formatting costs a given amount. If you suspect one, remove it, run again, and let the before-and-after decide.

Placement and clock: change where and when it runs

Lever	When it applies	Where
Split a monolithic job	Throughput is not scaling with node count	Distribute the load
Raise Tasks per Job	A single job's outputs are running more serially than the node can afford	Settings ▸ Configuration ▸ Performance ▸ Execution
Pin or exclude with Filter jobs	A long job is occupying nodes that serve short runs	Settings ▸ Load Balancer, per node
Stagger the schedule	The spike is a clock artifact, or a schedule is overrunning its own interval	Scheduled Content

Instance: change what the server does

Lever	When it applies	Where
Parallel Job Execution and Tasks per Job	The server has headroom, or is over-committed against its RAM	Settings ▸ Configuration ▸ Performance ▸ Execution
Enable Out of Process Job Execution	Should be on. Turn it off only on support advice	Settings ▸ Configuration ▸ Performance
Enable Out of Process Reporting Queries	Interactive add-in refresh is pressuring the web tier	Settings ▸ Configuration ▸ Performance ▸ Reporting
Retention and Compress History Files	History or stored output has grown large	Settings ▸ Configuration ▸ Performance, and Output retention
Debug Logging left on	Always worth checking. It produces very large logs and archives	Settings ▸ Configuration ▸ Logging
Recycle Service	A long-lived instance benefits from a daily restart	Settings ▸ Configuration ▸ Web Server

Contribution: change what a form costs

Lever	When it applies	What to do
Make the form smaller	Peak cost is high, and bytes per cell is normal	Cost is roughly bytes per cell multiplied by cell count, so removing an unused sheet or trimming to the range contributors actually fill removes real cost. Profile before and after.
Manual calculation mode	Large forms filled slowly over a day	The form stops recalculating on every edit and the server releases an idle session's workbook shortly after the last action that needed it, rebuilding on demand. The trade is that the contributor waits on their next action.
Max Sessions per Worker	Workers read **At capacity** while server memory is comfortable	The per-worker session cap is binding, not memory. Raising it lets sessions share a worker baseline, which lowers the per-contributor cost. Default **1**, range 1 to 50.
Recycle Worker After Sessions	High **Served (lifetime)** against a short **Uptime**	Workers are recycling often and contributors repeatedly pay a cold start. Default **25**; 0 never recycles.
Warm Worker Pool Size	Contributors wait on a cold start when they open a form	Keeps workers started and waiting. Default **0**, meaning no warm pool, so every session pays a start. Range 0 to 50.
Warm Worker Idle Timeout	Warm workers are being held longer than a round needs	How long an idle warm worker is kept before it exits. Default **30** minutes.

All four are on **Settings ▸ Configuration ▸ Performance**, under **Contribution**.

Full arithmetic and the live-round playbook are in Scale and size a contribution campaign.

Prove the change worked

An improvement you cannot demonstrate is an opinion. The measurement rules make a fair before-and-after straightforward, provided you know them.

- **The baseline is the median of up to 12 comparable earlier runs**, and the median is deliberate: one pathological run would drag an average far enough to hide every later regression.
- **A difference within 15 percent counts as noise** and is not flagged. If your change is worth less than that, this instrumentation will not show it, and neither will

your users notice it.

- **A run is never part of its own baseline**, and failed, canceled, and unmeasured runs are held out entirely.
- **Compare like with like.** Check the volume denominator on both sides. A faster run over less data has proved nothing.
- **Two different elapsed figures exist.** Where telemetry is present, elapsed is measured from execution start; where it is not, the column falls back to the wall-clock duration on the run record, and the tooltip says which you are looking at. They are never mixed inside one trend, and you should not mix them either.
- **Pin the runs that matter.** Performance data is removed when retention removes the run, so pin the before run if the comparison has to survive the retention window.
- **On contribution, deltas travel and absolutes do not.** Profiler numbers are working set on the machine that ran the profile. Compare before and after on the same machine.

Once several runs have accumulated after the change, the **Compare** tab should find the step you created, in the direction you intended. That is the cleanest evidence available.

Red flags and what they usually mean

Symptom	Usual cause	Where to look
Many reports regressed in the same week	A release or a node, not the reports	Settings ▸ Performance ▸ Distribution, the release and node pivots
One report drifted with no run ever flagged	A gradual step the dead band absorbed run by run	The Compare tab
Elapsed doubled and so did the volume	Not a regression	The volume denominator on the Performance tab
One engine dominates with a modest average	It is being re-run on every recalculation pass	The Engines table, **Recalc passes**
Occurrences are missing rather than late	A schedule overran its own interval past the 30-minute overdue window	Scheduled Content, and Distribute the load
Throughput did not improve when a node was added	A job runs atomically on one node	Distribute the load, job granularity
Analyst refresh is slow while batch throughput is fine	Interactive refresh is not distributed and lands on the node holding the session	Distribute the load
Contributors are refused a session	The 90% memory admission valve	Settings ▸ Performance ▸ Contribution
A column of dashes where measurements should be	Runs that predate telemetry, or runs held out of the baseline	The note under the run list, which states both counts

When to escalate

Reportworq's own surfaces attribute cost down to a phase and an engine. Below that, the support desk needs the raw material.

Collect it with **Settings ▸ Log Diagnostics**, which packages application logs from every node, including load-balancer nodes, into a support packet. Say which report and which **run** you are asking about, since a run's measurements are what support will read first. If you were asked to reproduce the problem with **Debug Logging** on, turn it off again after the capture: it produces very large logs and archives and is itself a drag on the instance. See Log diagnostics and support packets.

Notes and limits

- **This is not alerting.** There are no thresholds to set and no notifications. The baseline rules described here are fixed.
- **A failed run still records its performance,** deliberately, since a failure is often exactly the run whose breakdown you want. For diagnosing the failure itself rather than its cost, see Diagnose job failures.
- **Releasing a held report writes no new measurement.** The release re-opens the original run's record, and overwriting it would destroy the history this instrumentation exists to build.
- **Only one step change is reported** by the Compare tab, the single most significant one. It does not enumerate every wobble.
- **The report ranking on Settings ▸ Performance ▸ Distribution has no paging and no row cap.** On a large instance it is a long table, and narrowing the window is the only way to shorten it. A deleted report can still appear, because the page explains runs that already happened.
- **There is no comparison level below one variation** on the Compare tab. For per-workbook detail, use the Performance tab on a single run.

> **Going deeper.** For sizing and placement, see Distribute the load. For the full field-by-field reference on each screen, see Compare performance across reports, Review a run's performance, and Find when a report got slower. For the live contribution console, see Contribution processes.

Diagnose job failures

how-to administrator

When a job fails or completes with warnings, Reportworq keeps a per-run diagnostics archive you can read and trace. This guide is the administrator's path through those diagnostics, including how to send a run to support and the case where a job reports success but no output arrives.

When to use it. Whenever a run fails or flags warnings, or when a scheduled distribution completes but a recipient reports they got nothing.

Before you start

- Open the run from **Activity** (the Details container), or from a report's **Run History** by selecting **View diagnostics** on the run to open its "Report & diagnostics" view. Both host the same Log, Attachments, and Trace diagnostics.
- Any log requires that job-history logging was on for the run. If it was off, the diagnostics show "No execution log for this run," which points you to Report Properties ▸ Diagnostics to enable it before re-running.

Read the diagnostics

The **Diagnostics** view has three tabs:

- **Log.** Each entry shows a line number, timestamp, elapsed time, and message. For errors, the entry expands to the exception message and stack trace. The view shows the first 250 entries with a **Show Full Log** expander.
- **Attachments.** The files the run produced, viewable and downloadable.
- **Trace.** HTTP and HAR traces, useful for pinning down which datasource call failed. This tab is empty unless Trace logging was enabled for the run. To capture it, turn Verbose and then Trace logging on in the Run Job Options before re-running. See Log diagnostics and support packets.

When a job succeeds but produces no output

A run can report success and still deliver nothing. Work through these causes:

- **The exception check gated the send.** A report's exception check is a whole-report send gate. If no exception cell evaluated to the send condition, the job runs clean but sends nothing on purpose.
- **Every worksheet was suppressed.** Sheet suppression deletes a sheet when its formula is TRUE, and a provider report that returns no data auto-suppresses its

sheet. If all sheets suppress, there is no file to deliver.
- **A burst produced no recipients.** If the burstable parameter or burst set resolved to an empty recipient list, there is nothing to send.
- **The destination silently rejected it.** A Microsoft 365 email from a non-default address without a "Send as" grant, a Slack bot not added to the target channel, or a network folder the service account cannot write to, can all fail quietly. Check the destination setup and the run's Log tab.

Open the run's **Log** tab and read it end to end; a "sent 0" or a suppression entry usually names the cause.

Send a run to support

Sending a single run's own logs to Reportworq support is the fastest way to escalate one failure, because it packages exactly that execution's archive. This is the step customers most often get wrong, so work through it carefully. There are two entry points, and both upload the same execution logs.

From the run's history card. In a report's **Run History**, find the failed run and select **Send logs to support** on its card. (The same card also offers **Run again**.)

From the diagnostics view. Open the run's **View diagnostics**, then select **Send to support** in the toolbar. Use this when you are already reading the Log, Attachments, or Trace tabs.

Either way, Reportworq uploads that execution's diagnostics archive over the secure support-upload path and notifies the support desk. You do not attach anything by email, and you do not need to zip anything yourself.

If you need the archive as a file instead, select **Export logs** to download the run's execution database as a `.zip`. Use this to attach the archive to an existing support case, or when the instance cannot reach the support desk.

For a broader packet that spans more than one file, or one that collects logs from every server in a cluster, build a support packet instead. See Log diagnostics and support packets.

Notes and limits

- The Trace tab is empty unless Trace logging was on at run time, and Trace requires Verbose to be on.
- Enable Verbose logging before the run you want to diagnose, not after, so the detail is captured.

Going deeper. For report-author-side failure reading, see When a job fails. For symptom-to-fix pairs, see Troubleshooting index.

Output retention

how-to administrator

Retention governs how long a report's run history, its stored outputs and execution logs, is kept before older runs are removed automatically. It bounds storage growth while guaranteeing that pinned and policy-protected runs survive. This guide is for the administrator who sets those policies and understands what enforcement does.

When to use it. Whenever run-history volume must be controlled: set folder and report policies, pin what must survive, and let enforcement remove the rest.

How the policy resolves

Retention is a per-report and per-folder value, measured in days. For any given run, the effective policy is resolved along a chain, and the first level that carries a value wins:

1. **Output pin** on the run itself, which means keep forever.
2. **The report's** own retention value.
3. **The folder chain**, starting at the report's folder and walking up to the root; the nearest ancestor folder with an explicit value wins.
4. **The system default**, which is 30 days.

The day value follows one convention throughout: no value means inherit from the next level up, `0` means keep forever, and a positive number means keep that many days. When a single output is produced from more than one source report, the more conservative (longest-keeping) policy wins, so no report loses an output early.

Set a retention policy on a folder or report

1. Open the folder's or report's **Properties**, and find the **Job output history** section.
2. Use the **Inherit retention / Set a custom period** switch. **Inherit** uses the parent folder and then the system default. **Set a custom period** reveals **Keep output history (days)**, with a minimum of 1 day.
3. Save. Every report beneath a folder inherits the folder's value unless it sets its own.

For example, set the "Finance Reports" folder to keep run history 365 days, and every report under it inherits that unless it overrides.

Pin a run to keep it forever

Pinning is the only per-run exemption. A pinned run is kept forever regardless of any policy, and it is skipped by both enforcement and bulk delete.

1. Open the report's **Run History**.
2. On the run you want to protect, select **Pin**. Its effective label reads "Kept forever, pinned."

Use this for a final period-close run that must survive any future policy change.

Enforcement, what happens automatically

There is no "Enforce" button. On-premises, a hosted service ("Enforce Job History Retentions") runs automatically every hour. It walks the reports, skips pinned runs, resolves each run's policy, skips keep-forever policies, and marks a run expired when it is older than the resolved number of days, measured from the run's creation time. Expired unpinned runs are soft-deleted and then immediately purged to reclaim storage.

The Run History detail bar shows each run's effective retention, for example "Kept forever," "Kept until {date}," "Kept for N day(s)," or "Kept forever, pinned," along with the source of the policy (the report policy, a folder name, or the system default).

Delete a run, and recover one before purge

- **Delete** a run from Run History with **Delete run…**. This is a soft delete: the run leaves the active history list, but its output and artifacts remain on disk until they are purged.
- **Recovering a soft-deleted run** is possible only until it is permanently purged, and only as a Reportworq support task. There is **no restore button on Run History** (Run History offers **Delete run…** only), so if you need a soft-deleted run back, contact Reportworq support while it is still within the purge window. A run that has been hard-purged cannot be recovered.

The purge of soft-deleted items is a background store operation, not a button. Once purge removes a run, its payload, descriptor, and artifact files are gone for good.

Notes and limits

- The day convention treats `0` as keep forever, not delete immediately.
- Enforcement and purge run as background services on-premises, so removal is not instantaneous when you change a policy; the sweep runs hourly.
- Pinning is the only per-run exemption from a policy.
- Log and audit retention (application log, audit log, backup, AI audit, MCP audit) are governed by separate settings and are not the same as run-output retention. See Server configuration.

Going deeper. To review the durable record of what jobs ran and when, see Audit logs.

Licensing and entitlements

reference administrator

Reportworq is licensed through the Reportworq licensing service. The **License** tab on the **Settings > Configuration** screen is where an administrator views license status and details and activates, releases, or manually activates the license. The license carries **entitlements** that gate which features and connectors are available, so licensing determines not just whether the product runs but which capabilities appear.

The License tab

The License tab shows a license-status line, a license-info property list (an **Information** category and a **Details** category), and a masked key field. Unlike other Configuration tabs, it has no save state, **Save** stays disabled, because licensing changes happen through its own actions rather than the shared Save.

Action	What it does
Activate	Validates and binds the license against the Reportworq licensing service. Each activation consumes one **installation seat**, which is a seat for this installation of Reportworq. It is not a user assignment and has nothing to do with the role seats below.
Release	Reverts an activation and returns the installation seat, for example when moving or decommissioning an instance.
Manual activation	An offline request-code / response-code flow for a server that cannot reach the Reportworq licensing service directly. See Install Reportworq for the step-by-step.

The commercial model

A Reportworq license carries two kinds of thing: **role seats** for the people who sign in, and **installation-level capabilities** that turn features on for the whole environment. There is no limit on the number of reports you configure, and **recipients**, the people you distribute output to, are unlimited and free, because a recipient never signs in and holds no license. You license the people who *operate* Reportworq, not the reports or the audiences they reach.

Roles and seats

Reportworq licenses three roles. Each is sold as a seat count that is either numbered or unlimited, and each is drawn from its own pool. A fourth role, Recipient, needs no license

and is listed here for completeness:

Role seat	Who it is for
Administrator	Configures and manages the environment (security, users, integrations, content, licensing); reaches every workspace.
Author	Builds jobs (reports), data models, and campaigns within the workspaces granted to them; includes the Excel authoring add-in.
User	Signs in to consume Reportworq: view output by ACL, participate in contribution campaigns, refresh a distributed PowerPoint deck, or reach content through the MCP server and the AI assistant.
Recipient *(not licensed)*	Receives distributed output by email, Slack, Teams, SharePoint, and so on, without ever signing in. Holds no seat; unlimited and free.

Two naming changes matter when reading a license:

- The role that grants the full authoring experience now displays as **Author**. In earlier versions this same entitlement was labeled **Reportworq Administrator**. If a license note or older documentation says "Reportworq Administrator" as a user role, read it as **Author**.
- The **User** seat folds in several retired roles, **Reader**, **Consumer**, **Contribution End User**, and **PowerPoint End User** are all just a User now, and the older **Reporting End User** entitlement has been retired into **Author**.

For what each role can do, see Roles and what you can do.

Installation-level capabilities

Beyond the role seats, the license enables capabilities for the whole installation. These are properties of the license, not of individual users:

Capability	What it enables
Contribution	Contribution write-back, **including Data Collection**. When Contribution is licensed, Users can be assigned as contributors and approvers and collected data can be written back to a planning system. Data Collection is part of Contribution, not a separate line.
API	The **REST API** and the **Script Runner**. Without it, an API command or webhook call is refused with HTTP 400, "REST API not available for this license". The MCP server, CloudHub setup, and output downloads are not gated by this capability.
Load Balancer	A count of load-balancer nodes for a clustered deployment. **A count of 0 means load balancing is off.** See Deployment topology.
Data Connections	How many data-source connections may be configured, numbered or unlimited. Creating one while over the count is still allowed, but an overage stops jobs and campaigns from running. See the counting rule below and What happens when you go over a limit.
Workspaces	How many workspaces may be configured, numbered or unlimited. Creating one while over the count is still allowed, but an overage stops non-administrators from signing in. See What happens when you go over a limit.

If a counted limit is missing from the license, the default differs by lever, deliberately: **Load Balancer** defaults to **off**, while **Data Connections** and **Workspaces** default to **unlimited**. A feature that has to be bought is never given away by an unstamped license, and a limit on something every installation already has never cripples one.

When a feature is unavailable, check the License tab before requesting an upgrade. For example, without Contribution the Data Collection settings show a "not included in your license" lock, and Reportworq hosted AI is filtered out of the integrations catalog without the hosted-AI entitlement.

How data connections are counted

Every configured data-source connection across **all** workspaces counts as **one** connection toward the Data Connections limit, with a single exception: the **Excel file** source (the Excel/CData connector) is **free and does not count**.

> **Worked example.** An installation with 3 IBM Planning Analytics systems, 2 SQL Server databases, and 3 Excel file sources uses **5** counted connections. The 3 Excel files are free, even when spread across several workspaces.

How seats are counted

- Each role, Administrator, Author, and User, draws from **its own** seat pool. An Administrator seat is not an Author seat, and an Author seat is not a User seat.
- **An Administrator is the full-access user and can do everything an Author can**, including building jobs, data models, and campaigns. Authoring as an administrator consumes an **Administrator** seat, not an Author seat. So one Administrator plus one Author is two seats in total, drawn from two different pools, and the Administrator is the one who can also configure the environment.
- At least one seat is required to sign in, and the instance must always keep at least one Administrator. If you remove every administrator you will lock yourself out, see the lockout-recovery note in Sign-in and SSO.

Non-production licenses

You can request a **non-production license key** for development and test environments. It is fully functional, so a dev or test instance behaves exactly like production, with two deliberate differences that keep non-production output from being mistaken for the real thing:

- Reports carry a **watermark**.
- The user interface shows **non-production notices**.

Ask Reportworq for a non-production key rather than pointing a second environment at your production key. Each activation consumes an installation seat, so a shared key would draw a seat you meant to keep for production.

What happens when you go over a limit

Going over a licensed limit never fails at the moment you cause it. You can assign one more entitlement, add one more data connection, or create one more workspace while already over. **The enforcement happens later, and it is real:** an overage is not a cosmetic notice on the License screen.

Sign-in is blocked while you are over the User seat count or the Workspace count. An **Author** or a **User** who is not an administrator is refused at the portal with a message naming the User or Workspace limit. **Administrators are exempt**, deliberately, so an administrator can always get in to disable accounts or remove a workspace and bring the installation back under its limits. The Excel add-in task pane is also exempt.

Jobs and campaigns fail to run while you are over a count or a seat limit. A **Data Connections** or **Load Balancer** overage is checked when the run is queued, and

the run then fails with the reason recorded against it. **Administrator** and **Author** seat overages are checked as the run starts and fail the same way. This check is deliberately fail-open: if the licensing service cannot be reached, the job is allowed to run rather than blocked by an unrelated fault.

The API capability is a separate, immediate block. Without it, REST API commands and webhooks are refused outright, whatever your counts are.

A limit only ever enforces when the license carries a finite value **and** the live count exceeds it, so a license that has not been reissued does not suddenly start blocking.

> **Watch the counts before they bite.** Because nothing fails at assignment time, the first symptom of an overage is usually a user who cannot sign in or a scheduled job that stopped running. Check the License screen when either happens.

This matters most with single sign-on. An entitlement granted through an identity-provider group only draws a license when each account first signs in, so an overage can surface gradually as users come online. To avoid it:

1. Assign entitlements at the **group** level rather than per account, so counts are easy to reason about.
2. Keep any OIDC-provider group that is mapped to a feature **under that feature's licensed count**.
3. Pre-provision accounts and reconcile against seat counts before a rollout.
4. Run periodic user and license audits.

See Accounts, groups, and entitlements for the entitlement catalog and the lever to release a seat after removing a role.

Notes and limits

- The License tab reaches an external service, and activation consumes an installation seat, so it is not exercised casually.
- Licensing events (activation and validation) are recorded in the System audit log under the **Licensing** category, see Audit logs.
- Load balancing is licensed as the **Load Balancer** node count, enabled per license on request and verifiable on the License screen; a count of 0 means it is off, see Deployment topology.

> **Going deeper.** For moving a license between servers, release on the old server and activate on the new one, see Migrate to a new server.

Workspaces

how-to administrator

A Reportworq server can host **multiple workspaces**, separate content spaces that each have their own folder tree, items, data connections, and security scope. One sign-in moves between them, so a single server can segregate content by environment, business unit, or tenant.

When to use it. Whenever a deployment needs more than one isolated content space, for example a workspace specific to an organizational company or department. A single-workspace deployment needs none of this and shows no switcher.

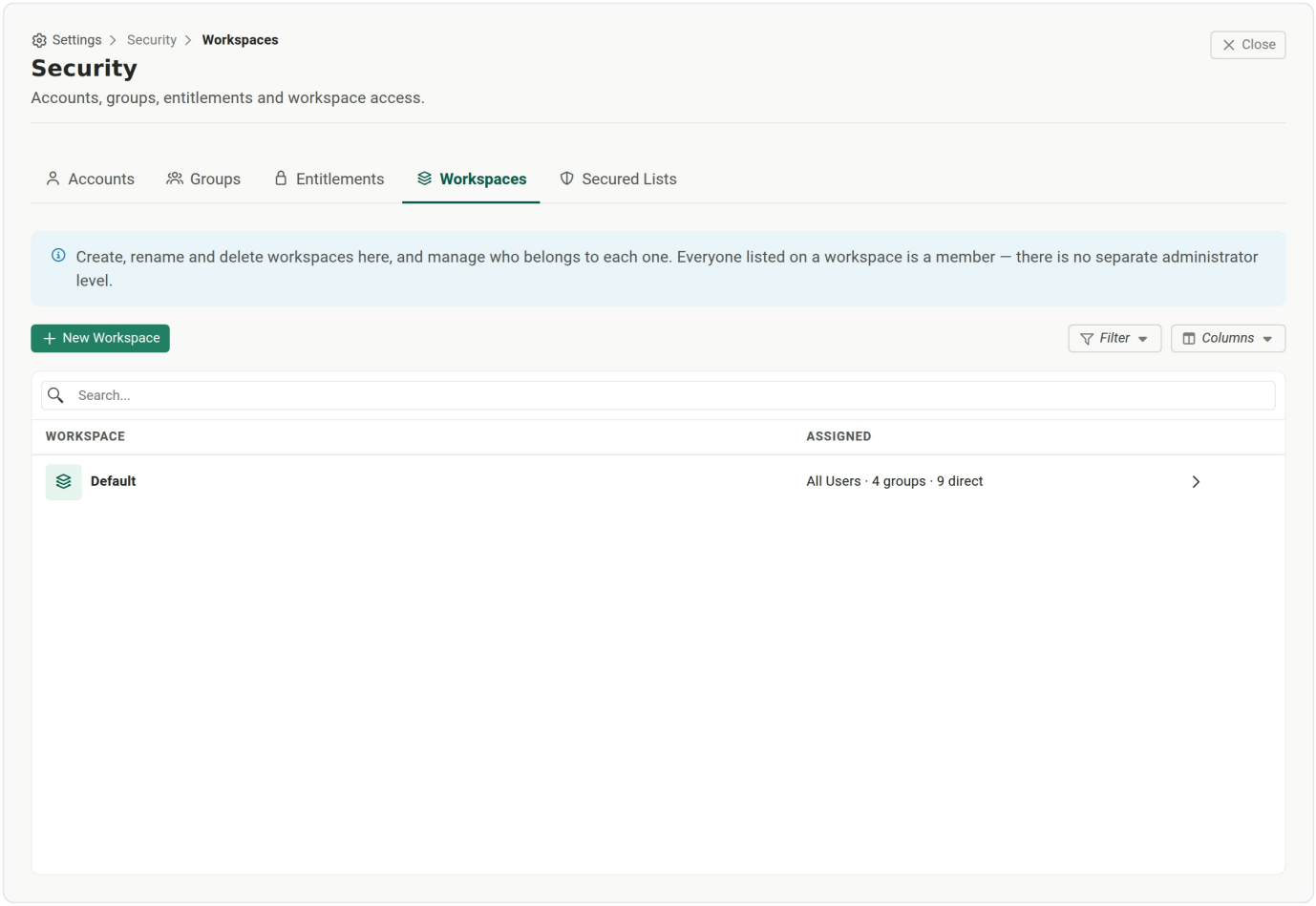
Before you start

- You need Administrator rights to create, rename, or delete a workspace and to grant access to one.
- Everything about a workspace is managed in one place, **Settings ▸ Security** on the **Workspaces** tab: the workspaces themselves, and who belongs to each one.

> **Looking for workspaces under Configuration ▸ Web Server?** Workspace creation and deletion used to sit in the Configuration screen's **Web Server** tab. That category has been removed, and the whole workspace story now lives on **Settings ▸ Security ▸ Workspaces**.

Create, rename, or delete a workspace

1. Open **Settings ▸ Security** and select the **Workspaces** tab. The list shows one row per workspace, with the number of groups and accounts assigned to each.
2. To add one, select **New Workspace**, enter a name, and confirm. The new workspace appears in the list.
3. To rename one, select its row to open it, then select the title and enter the new name. The change is saved when you commit the title.
4. To delete one, select its row and then select **Delete Workspace**. The confirmation states how many jobs and how many schedules the workspace holds, because deleting it makes all of that content inaccessible. Read the counts before you confirm.



Manage who belongs to a workspace

Everyone listed on a workspace is a **member**, and membership is the only level. A member can enter the workspace and reach its content, as their per-item permissions allow.

1. On the **Workspaces** tab, select a workspace to open it.
2. Assign the groups and accounts that need it. Turning on the **All Users** grant gives every enabled account access instead.
3. To work from the other direction, open an account or a group and use its **Workspaces Granted** section. The two routes edit the same grant.

An account can be a member of several workspaces. Workspace membership is layered on top of the installation-wide role (Administrator, Author, or User), which does not vary by workspace.

An **Administrator** has full access to every workspace regardless of any grant, and sees the switcher as soon as a second workspace exists. The account editor says so directly: *"Administrators have access to all workspaces."*

> **There is no per-workspace administrator.** A **Configuration Access** level, sometimes called Workspace Administrator, existed in earlier versions and has been removed. Settings, including datasource configuration, is now **Administrator-only**, with no limited or partial access. Workspace management cannot be delegated below the Administrator role.

How a user switches workspace

A user with access to more than one workspace switches from the **avatar menu** in the app header, in the **Switch-Workspace** group. Selecting a workspace confirms with a dialog, then reloads into the target workspace, and the content browser and filters follow the new active workspace.

For an account with access to only one workspace, the Switch-Workspace group is absent, so the switcher only appears when it is relevant.

> **One active workspace per browser tab.** A user works in a single active workspace per tab, and the switcher defaults to the last-accessed one. To work in two workspaces at once, open separate tabs or windows.

What is shared and what is per-workspace

Some configuration is instance-wide, and some lives independently in each workspace:

Shared across all workspaces	Unique per workspace
Authentication (the active provider plus accounts, groups, and entitlements)	Jobs
Server configuration	Contacts
Microsoft 365 integration	Global variables
Cloud API	Schedules
Distributor configuration	Datasources and report providers

So an account and the authentication provider are set once for the server, while the reporting content and its data connections live independently in each workspace.

The shared job queue

> **All workspaces share one execution queue, and other-workspace rows are redacted.** The Running Jobs and Activity screen shows jobs from every workspace so you can see the whole queue. But you cannot expand, cancel, or even see the name of a job that belongs to a different workspace, those rows are redacted. To interact with such a job, switch to that job's origin workspace first.

Notes and limits

- Single-workspace deployments have no switcher and need none.
- Workspace membership decides whether an account can enter a workspace. Which specific items inside it the account can see is governed by per-item access, and datasource configuration is Administrator-only. See Roles and what you can do.
- The Local REST API key is separate per workspace (the key selects the workspace), whereas the Cloud API uses one global key plus a `?workspace=` selector.

> **Going deeper.** The per-workspace roles you assign here reference the accounts and groups defined in Accounts, groups, and entitlements.

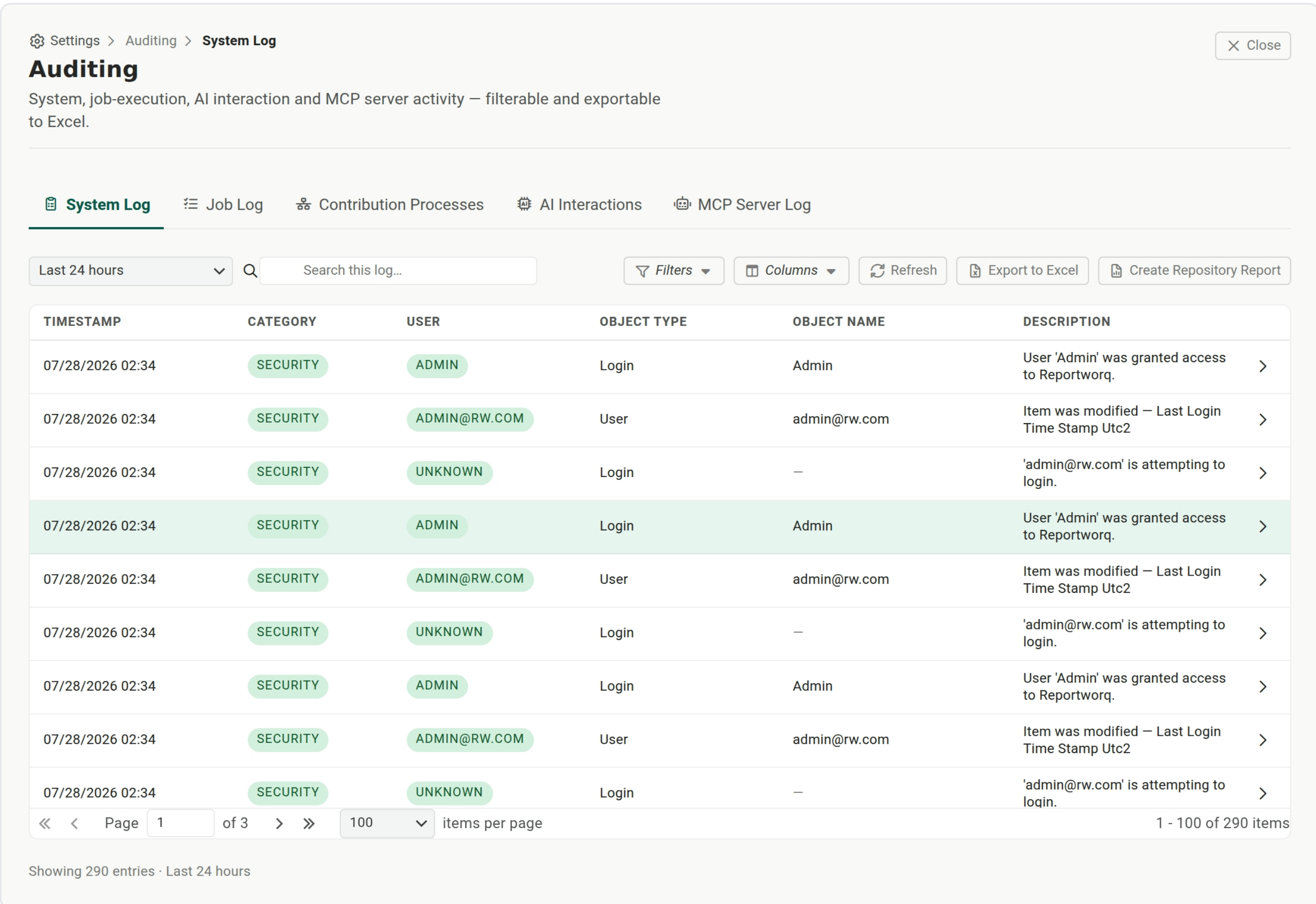
The repository metadata report

reference administrator

The **Repository metadata report** is a single-click Excel export that documents a workspace's entire distribution configuration. It is the fastest way to inventory a large library, answer an audit question, or plan a migration, because it turns configuration that is otherwise spread across many screens into filterable spreadsheet columns.

Generate it

1. Open **Settings ▸ Auditing**.
2. Select the **System Log** tab.
3. Select **Create Repository Report** in the toolbar. A single action produces and downloads the `.xlsx` file, named `Repository Report_{date}.xlsx`.



There is nothing to configure. The report covers the workspace you generate it in, as that workspace stands at the moment you generate it. On an instance with several workspaces, generate one report per workspace and switch workspaces between runs.

The seven worksheets

The workbook has **seven** worksheets, each a table of one configuration object type. Every sheet except **Contacts** opens with a **Workspace** column naming the workspace the row came from, so exports from several workspaces can be stacked into one sheet and still be told apart.

Worksheet	What it lists
Jobs	Every distribution job, with columns including `Is Scheduled` and `Distribution type`.
Reports	Every report, with columns including the source workbook `File Path`.
Report Parameters	The parameters attached to each report.
Parameters	Parameter definitions, with a `Parameter Type` column.
Static Burst Sets	The static burst sets defined in the instance.
Schedules	The schedules and what they run.
Contacts	The address-book contacts.

Detail worksheets carry embedded cross-navigation hyperlinks that jump back to the related row on the **Jobs** sheet (and from Report Parameters back to **Reports**), so you can trace configuration relationships without matching IDs by hand.

Use it as an audit tool

Because every column is a plain Excel column, you can filter and pivot the workbook to answer questions that no single screen answers directly:

- **Find every job on a given delivery channel.** Filter the **Jobs** sheet by `Distribution type`, or by `Is Scheduled` to list only scheduled jobs.
- **Find every job using a parameter kind.** Filter the **Parameters** sheet by `Parameter Type`, for example to list everything that uses IBM Planning Analytics subsets.
- **Enumerate every referenced workbook.** Extract the distinct `File Path` values from the **Reports** sheet to list every Excel workbook the instance's jobs depend on. This is invaluable for co-location and migration audits.
- **Inventory recipients.** Read the **Contacts** sheet for the full contact directory in one table.

When to use it

- **Before a migration**, to enumerate every referenced source workbook so nothing is left behind, see Migrate to a new server.
- **During an audit**, to produce evidence of what runs, on what schedule, to which channels.
- **When taking over an unfamiliar instance**, to understand the whole distribution configuration at a glance.
- **When troubleshooting scope**, to see every job affected by a shared parameter or datasource before you change it.

Notes and limits

- The report is a point-in-time snapshot. Regenerate it after configuration changes.
- The report is scoped to one workspace. The **Contacts** sheet is the exception to the Workspace column, so label it yourself if you combine contact exports from several workspaces.
- It documents distribution configuration (jobs, reports, parameters, burst sets, schedules, contacts), not run history or logs. For those, see Audit logs.
- The report describes an instance; it does not restore one. It is a standalone Excel export, separate from the Configuration backup zip.

> **Going deeper.** For the audit log tabs on the same Auditing screen, see Audit logs.

Compare performance across reports

how-to administrator

A report's own **Run History** answers "is this report slower than it used to be". It cannot answer the two questions an administrator asks when several things feel slow at once:

- **Did a release regress us?** Something changed after an upgrade, but no single report proves it.
- **Is a node slower?** One machine in a cluster is doing worse work than the others, and no report's own history can see across machines.

The **Distribution** tab of **Settings ▸ Performance** answers both. It reads the same run measurements the per-report surfaces read, but scoped across every report instead of within one.

Settings ▸ Performance has two tabs. Distribution, described on this page, covers report and job execution. **Contribution** is the live monitor for contribution worker processes and sessions, described in Contribution processes. The two are separate surfaces on one screen; nothing on this page applies to the Contribution tab.

When to use it. After an upgrade, when several users report slowness in the same week, when a cluster node is suspected, or as a periodic check on which reports are drifting.

Before you start

- You need the **Administrator** entitlement. The page is part of Settings, which is administrator-only and is not offered on a load-balancer node.
- The page reads run history. Only runs executed on a build that records performance telemetry carry measurements, so an instance that has just upgraded will have a thin page until new runs accumulate. See Review a run's performance for what a run records.

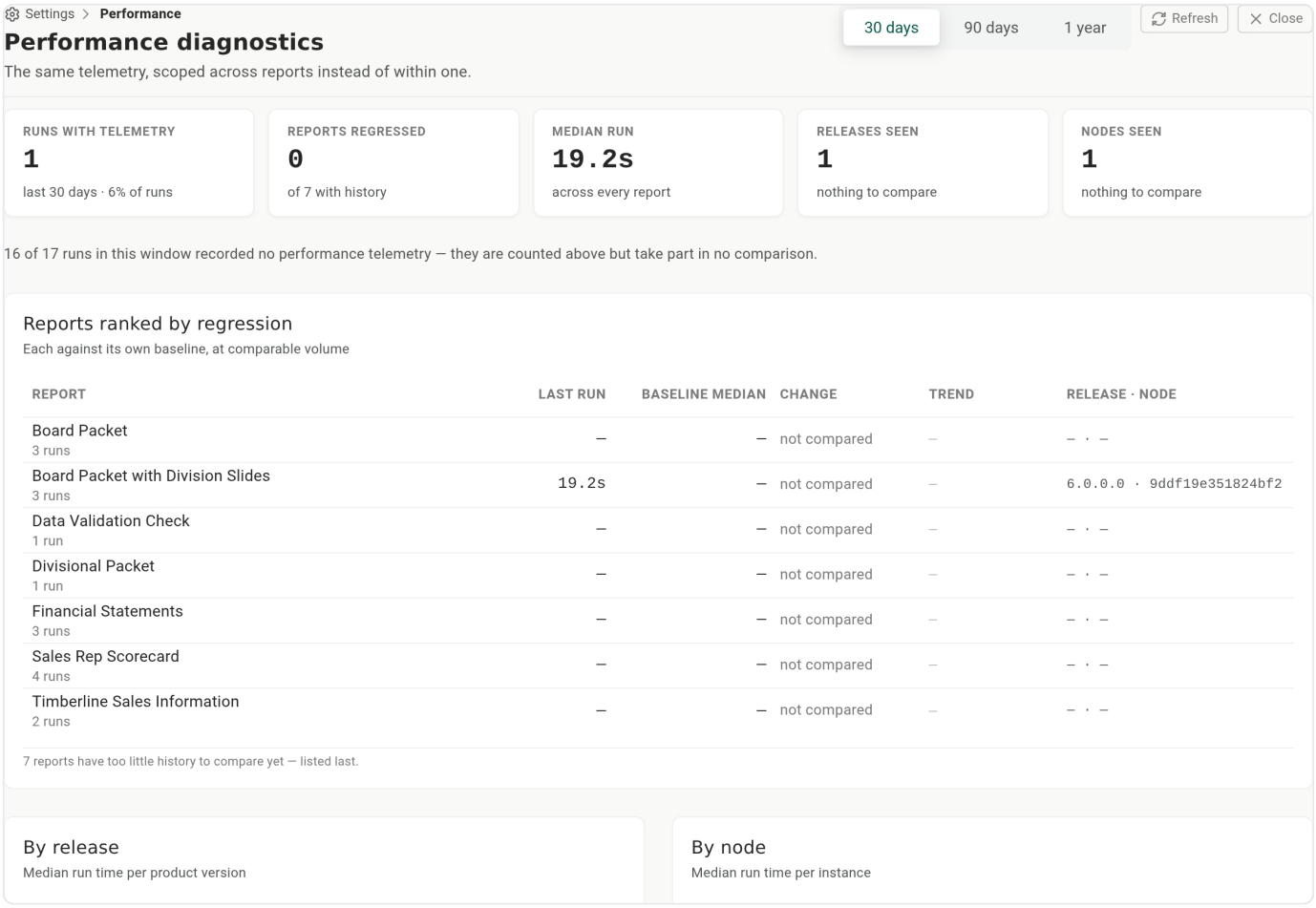
Open the page and choose a window

1. Open **Settings**.
2. Select the **Performance** card. Its description reads "Distribution and contribution: which reports are slowing down, whether a release or a node regressed you, and live contribution workers and sessions."
3. The page opens on the **Distribution** tab. The breadcrumb reads **Settings › Performance › Distribution**, and the address bar ends in `/distribution`, so the tab you are on is part of the link you can share or bookmark.

4. Choose the window: **30 days** (the default), **90 days**, or **1 year**. Everything on the page, including the baselines, is recomputed inside the window you choose, so a delta on screen is always explained by runs you can see.
5. Select **Refresh** to re-read run history. Runs written by an out-of-process job runner reach the web application as a notification, and Refresh is what you use if a run you expect is not there yet.
6. Select **Close** to return to Settings.

The window selector and **Refresh** belong to the Distribution tab and are absent while the Contribution tab is showing; that tab carries its own auto-refresh switch and its own Refresh.

If nothing has run in the window, the page says "No runs in this window." and suggests widening the window or checking that job-history logging is on.



Read the five figures

Figure	How to read it
Runs with telemetry	How many runs in the window carried a measurement, with the window length and the percentage of all runs beneath it. A low percentage means most of this window predates telemetry, and the page below is thinner than it looks.
Reports regressed	How many reports are more than 15 percent above their own baseline, out of every report with run history in the window.
Median run	The middle run duration across every report. It is taken over successful runs only, so a batch of early failures cannot make the instance look faster than it is.
Releases seen	How many distinct product versions ran in the window. One release is "nothing to compare".
Nodes seen	How many distinct instances ran in the window. One node is "nothing to compare".

Where some runs in the window recorded nothing, a line under the figures says how many. Those runs are counted in the totals but take part in no comparison.

Read the report ranking

Reports ranked by regression lists reports worst first, each measured against its own baseline at comparable volume.

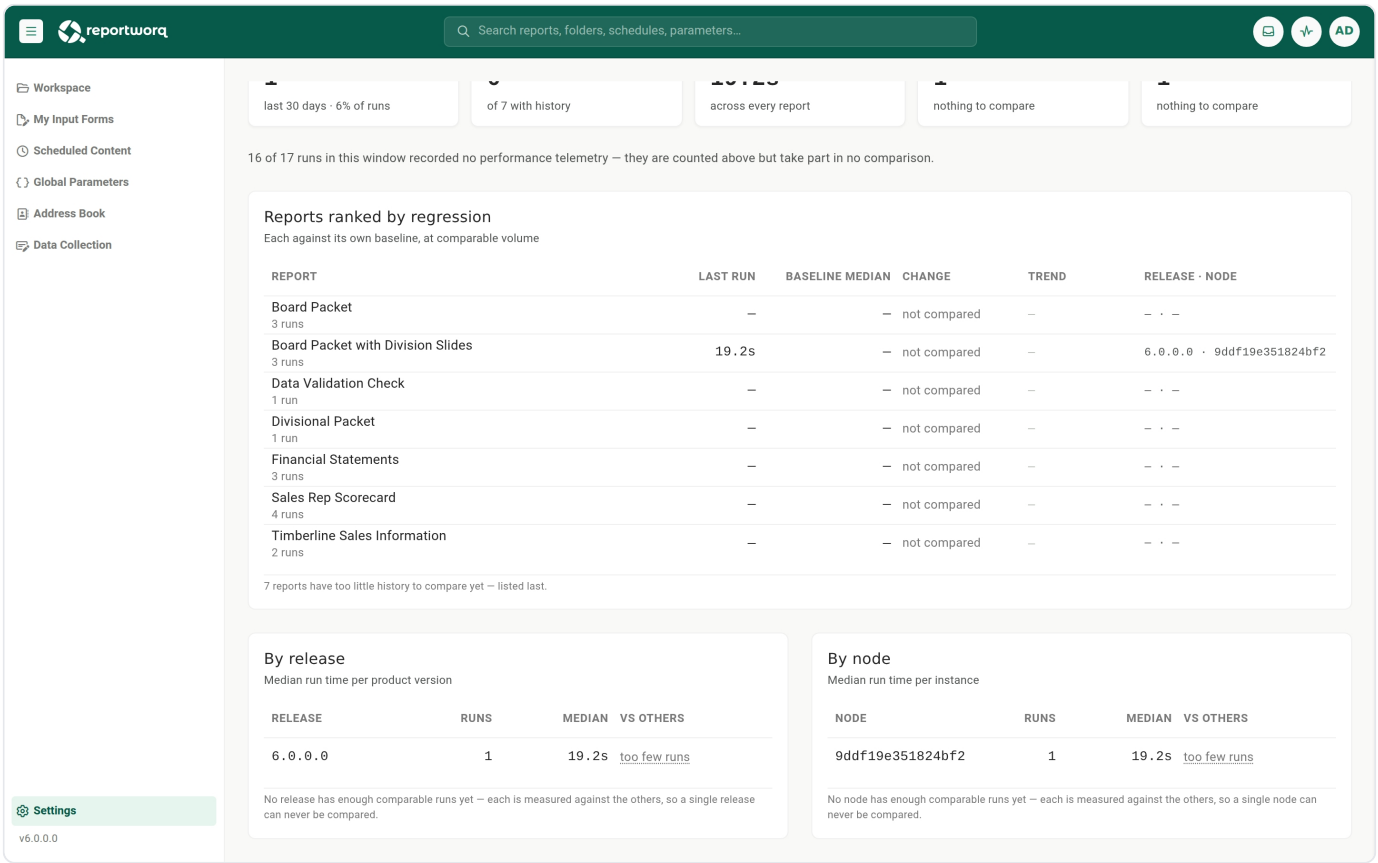
Column	What it shows
Report	The report name, with how many runs it had in the window.
Last run	The elapsed time of its most recent run in the window.
Baseline median	The median of that run's comparable predecessors.
Change	The difference between the two, as an up or down indicator. Where no fair comparison is possible the cell reads "not compared" and the reason is in its tooltip.
Trend	The report's run times in the window as a small chart. A run with no measurement breaks the line rather than dropping it to zero.
Release, node	The product version and the instance name of the most recent run. On a single-server installation the instance name is the server's own generated instance id, so the useful reading here is "did this change", not the name itself.

The baseline rules are the same ones the per-report run list uses: the median of up to 12 comparable earlier runs, at least 3 of them before any comparison is offered, a 15 percent dead band, a run is never in its own baseline, and failed, canceled, and unmeasured runs are held out. They are described in full in Review a run's performance.

Reports that cannot be compared are listed last rather than dropped, and a note under the table says how many. "We could not compare this" is information too.

Read the release and node pivots

Two tables sit side by side at the foot of the page: **By release** (median run time per product version) and **By node** (median run time per instance). Both have the same columns. On a narrow browser window the two tables scroll sideways inside their own panels rather than pushing the page wider, so the last column may need a sideways scroll to reach.



Column	What it shows
Release or **Node**	The product version, or the instance name.
Runs	How many comparable runs the group contributed.
Median	The group's median run duration.
vs others	How the group compares to every run outside it.
Peak memory	The highest working set any run in the group reached.

Three rules govern these pivots, and they are why a cell sometimes refuses to give you a number.

- **A group is compared against the other groups, never against the overall median.** A node that runs most of the work would drag an overall median toward itself and look normal no matter how slow it was.
- **A group needs at least 3 runs** before it is compared, and the cell reads "too few runs" below that. There also has to be something to compare it with: a single release or a single node can never be compared, and the note under the table says so rather than leaving a blank cell unexplained.
- **Volume has to match.** If a group processed more than 15 percent more or less data than the others, the cell reads "different volume" and no verdict is given. A release that looks 40 percent slower because its runs happened to carry three times the data is not a regression, and reporting it as one would make every other row untrustworthy.

Only successful runs take part. A run that failed or was canceled has a real duration and no meaning, and letting it into a median turns a crash-heavy node into a fast one.

Notes and limits

- **The report table has no paging and no row cap.** It lists every report with a run in the window. On a large installation that table is long, and narrowing the window is the only way to shorten it.
- **Runs, not reports, define the list.** A report that has been deleted still has run history, so it can appear here. That is deliberate: the page exists to explain runs that already happened.
- **It reads only the summary carried on each run's history entry.** No execution database and no performance artifact is opened, which is what makes the page usable on an instance with thousands of runs. The cost of that choice is that nothing below whole-run totals is available here. For a breakdown by output, workbook, phase, or calculation engine, open the run itself, see Review a run's performance.

- **This is not alerting.** There are no thresholds to set and no notifications. The window selector and Refresh are the only controls on this tab.
- **Performance data ages out with the run.** It is stored with the run and is removed when retention removes the run, see Output retention.

> **Going deeper.** For the section this page belongs to, including how to size a deployment and the levers worth pulling once you have found the problem, start at Performance and capacity planning. To break a single run down until the cost is attributable to a phase or an engine, see Review a run's performance. To find the point in a report's history where its behavior changed, see Find when a report got slower. For the other tab on this screen, see Contribution processes. For collecting logs for the support desk, see Log diagnostics and support packets.

Update Reportworq

how-to administrator

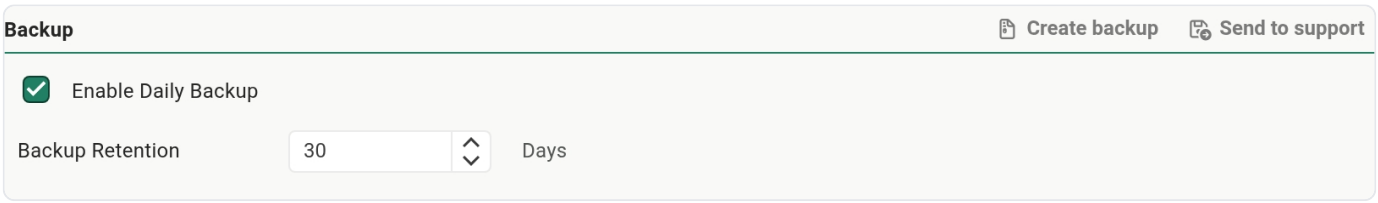
Settings > Update Reportworq is the in-product way to keep an instance current. It lists available software versions and updates the instance to the latest official release, the latest hotfix, a specific version, or from a patch file. Update actions download and install builds and restart the service, so they are administrator-only and should run in a maintenance window.

Before you begin

- System-administrator rights.
- A maintenance window, updates restart the service and interrupt active reporting.
- For online methods, connectivity to the update source. For an isolated server, a patch file provided out of band.
- A backup, see below.

Back up before you update

1. Create a timestamped backup from **Settings > Configuration > Web Server > Backup**, then select **Create backup**. Reportworq writes a dated archive (for example `reportworq_data_backup_2024-10-05 20.41.38.zip`).
2. Note the Bootstrap Service startup type and logon account.



Modern upgrades also auto-back-up the Repository before applying. Keep the pre-upgrade backup: occasionally an upgrade introduces non-backward-compatible database changes, so downgrading past such a build is unsafe.

If you need to downgrade, or a downgrade does not behave as expected, contact Reportworq support before going further. Reversing a version that changed the Repository is not a self-service operation, and support can tell you whether the build you are on is one of the affected ones.

Restore from a backup

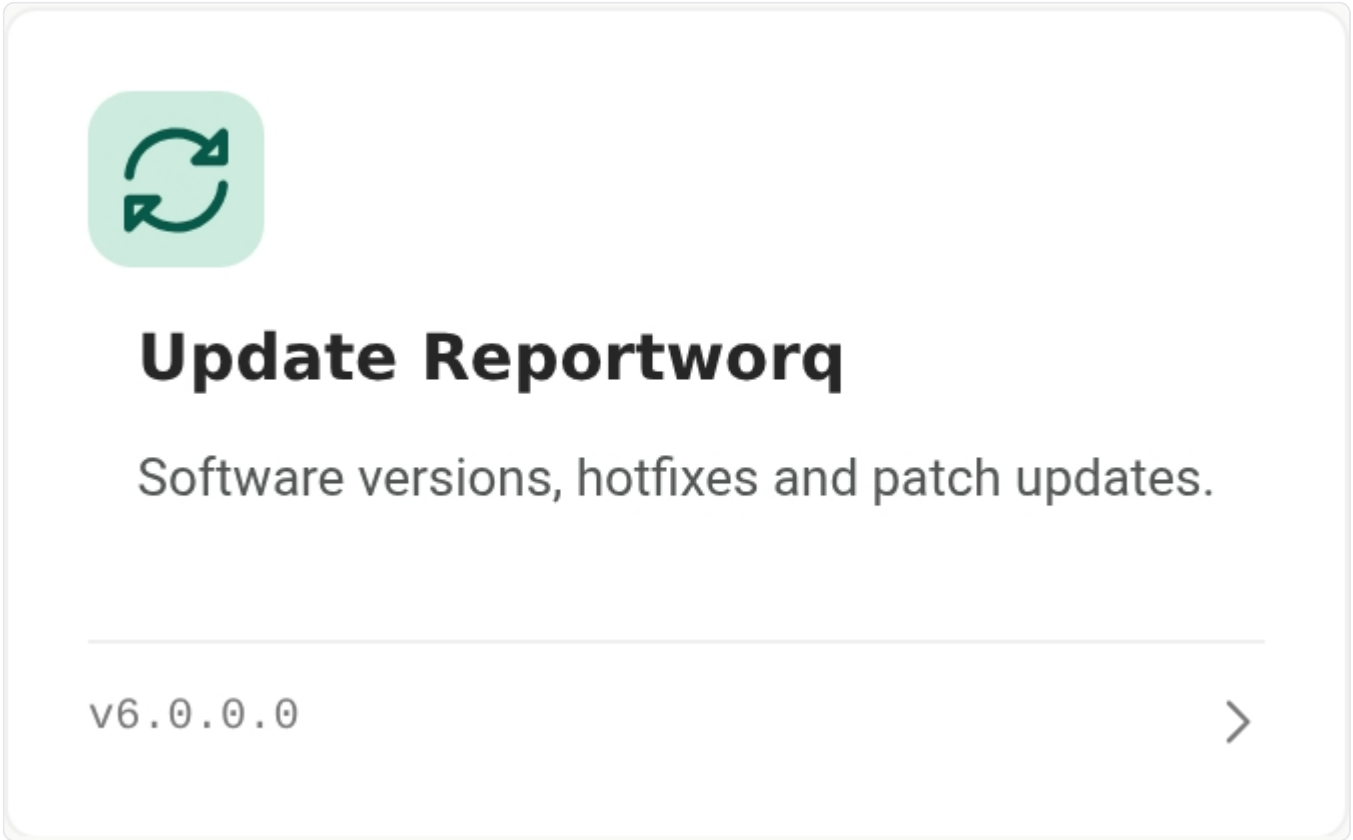
Reportworq has no in-product restore button; a restore is a manual file operation. Do it with the service stopped so no process holds the Repository open, the same discipline

used when moving servers.

1. Stop the **Reportworq 6 Bootstrap Service** on the server (`services.msc`).
2. Extract the backup archive's contents into the **Repository folder**, replacing the current contents.
3. Start the Bootstrap Service.
4. Sign in and confirm jobs, schedules, and datasource connections are intact.

Update Reportworq

1. Open **Settings > Update Reportworq**. The versions grid loads in the background; use **Refresh** if it is empty or slow.
2. To see hotfix builds, turn on **Show Hotfix Versions**. A hotfix is a fully tested update to the current official version that contains only critical fixes, no new features and no experimental changes. Hotfixes are hidden by default and applied on support's guidance.
3. Open the **Update** menu and choose a method:
    - **Update to latest official version**, installs the newest official release.
    - **Update to latest hotfix version**, installs the newest hotfix build (shown only when hotfixes are visible).
    - **Download a specific version**, targets a chosen version from the grid.
4. The update installs and restarts the service. Confirm the new version on the **Settings** landing card afterward.



Update an air-gapped instance

When the server cannot reach the update source, update from a patch file provided out of band.

1. On an internet-connected machine, download the patch file (`Reportworq_Patch_<version>.zip`, for example `Reportworq_Patch_5.0.0.75.zip`) for the target version.
2. Transfer it to the server.
3. In **Settings > Update Reportworq**, open the **Update** menu and choose **Update from a patch file**, then select the transferred file. The upload shows a progress indicator while it applies.
4. The update installs and restarts the service. Confirm the new version on the **Settings** landing card.

Update a container deployment (Docker or Azure App Service)

A container deployment runs from a **read-only image**, so it does not update itself. The steps above do not apply, and the product will tell you so rather than letting you try.

On these deployments the **Update Reportworq** screen becomes **informational**:

- The **Update** menu and every per-row activate, remove, and download action are hidden.
- A callout reports the **running version** and how to change it.
- The version grid stays visible, read only, so you can still see what is running and what exists.

The update itself happens at the container host, not in Reportworq. Swap the container image tag, or the App Service container setting, to the version you want and restart. The new image is the update, and the Repository on its mounted storage carries your content across unchanged.

Update worker nodes the same way, and keep them on the same version as the web instance.

> **Why it works this way.** A container's filesystem is meant to be disposable, so an in-app updater that rewrites the install directory and restarts a service would fight the platform. Reportworq detects the immutable deployment profile and steps aside. See Deployment topology.

Update a load-balanced cluster

All servers in a cluster should run the same Reportworq version, so update them together in a maintenance window.

1. Update the main server first, using the steps above.
2. Update each load-balancer node to the same version.
3. Coordinate the node restarts so jobs are not dispatched to a node mid-update. On the Load Balancer screen, a node's Status reflects whether it dequeued a job in the last two minutes; let nodes settle before and after each update.

See Deployment topology for the cluster model.

Notes and limits

- Update actions are disruptive (install plus restart) and should not run during active reporting.
- Software-version and restart events are recorded in the System audit log under the **Informational** category, see Audit logs.

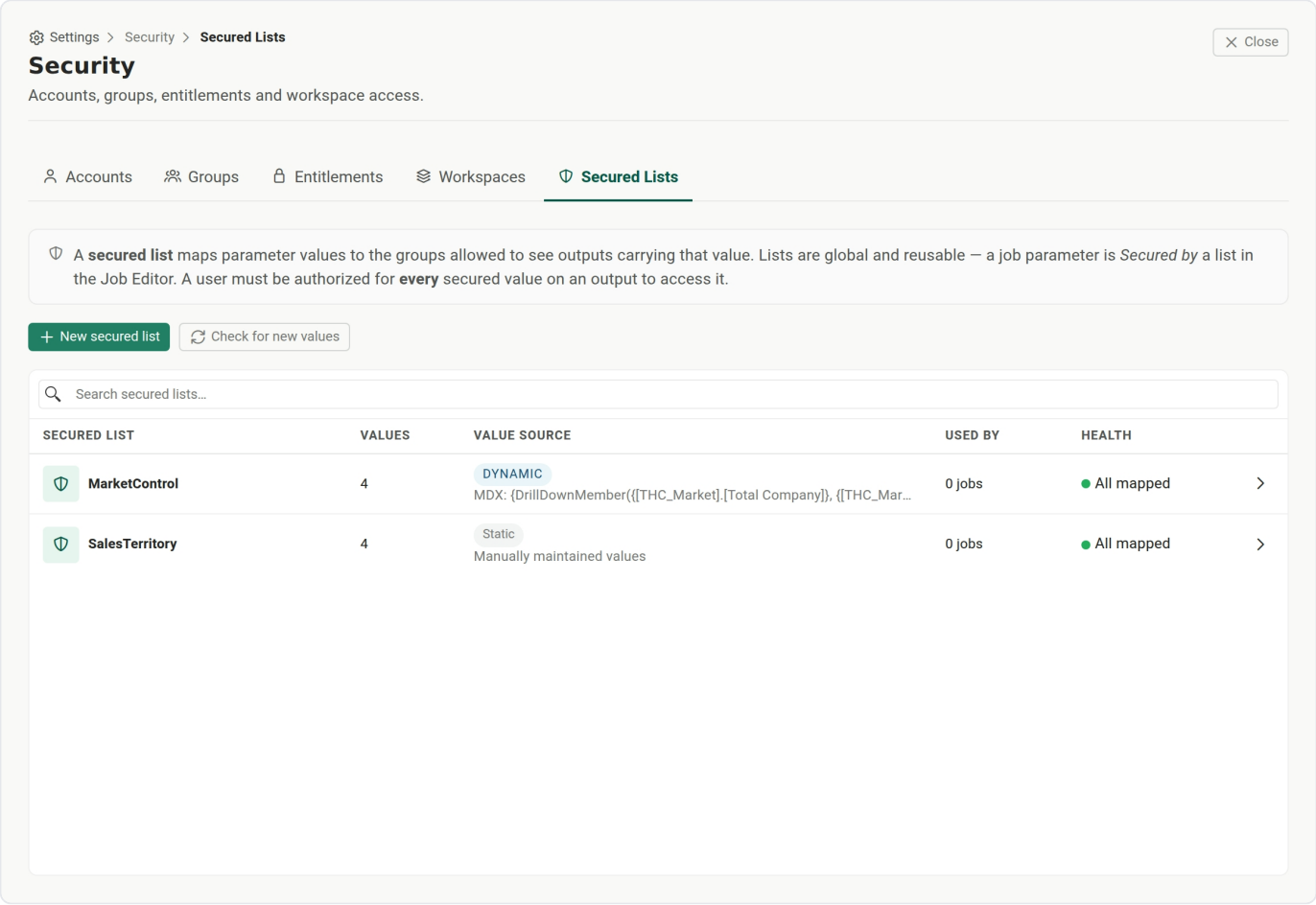
> **Going deeper.** The backup-and-restore-into-Repository discipline here is the same one used when moving servers, see Migrate to a new server.

ADMINISTRATOR GUIDE · SECURITY & ACCESS

Secured lists

how-to administrator

A **secured list** is a named mapping from a set of values to the security groups allowed to see the report outputs associated with each value. It is the mechanism that decides **whether a User can see a bursted output**: when a job is bursted on a secured parameter, each output is tagged with a value, and the secured list decides which groups, and therefore which Users, may open that output. Secured Lists is the fifth tab on the **Settings ▸ Security** screen, alongside Accounts, Groups, Entitlements, and Workspaces.



> **Secured lists are report-level, not row-level.** A secured list controls whether a User can open a whole **bursted output**, not which rows a User sees inside a report. Adding a User to a group does **not** hide other rows, markets, or regions within a report the User is allowed to open: whoever can open a report sees all of its rows. To restrict what each audience sees, burst the report on the secured parameter so each audience gets its own output, then map those output values to groups here.

> **Secured lists apply to the User role only.** Authors and Administrators see all output regardless of any secured list, and Recipients never sign in, they simply receive whatever a distribution delivers to them. Use secured lists to scope what signed-in **Users** can open, not to control who a distribution sends to.

When to use it. When governed data must be sliced by a dimension (region, cost center, entity) and each slice restricted to a specific group, so one report bursted on a secured parameter can be safely distributed to many audiences without any of them opening another's output.

Before you start

- You need administrator rights.
- The security groups you will map values to must already exist. Create them first on the **Groups** tab. See Accounts, groups, and entitlements.

Create a secured list and map values to groups

1. Open **Settings ▸ Security** and select the **Secured Lists** tab. It shows a grid of existing lists, or an empty state reading "No secured lists yet."
2. Select **New** and enter a name. Reportworq drills straight into the new list's detail. A new list defaults to **Static**.
3. On the list detail, select **Add value** and enter a value, for example a region code. It appears as a row in the value-to-groups mapping table.
4. For each value, map the security groups permitted to see the bursted output carrying that value.
5. Use **Rename** to change the list name, or **Delete** to remove the list (a destructive confirmation appears). Use the header **Close** to go back to the grid.

Static and dynamic lists

- A **static** list holds hand-entered values. Its detail shows **Rename**, **Delete**, and **Add value**, and does not show a **Refresh** control.
- A **dynamic** list discovers its values from a source rather than by typing. It shows a **Refresh** control to re-pull values, and the toolbar-level **Check for new values** scans for values that have appeared in the source but are not yet mapped.

Assign newly discovered values

1. When new dimension members appear in the source, for example new cost centers, select **Check for new values** on the Secured Lists toolbar.

2. Assign each newly discovered value to the correct group before the next distribution, so no output ships with an unmapped value.

Notes and limits

- Secured lists are **report-level**. They decide whether a User can open a whole bursted output. They do not hide individual rows within a report (whoever can open a report sees all of its rows), control item-level access to a report (that is workspace security), or set distribution recipients (that is contacts and the address book).
- Secured lists apply to the **User** role only. Authors and Administrators see all output; Recipients never sign in.
- Retire a value by deleting its mapping, or the whole list, when a slice is consolidated, so stale grants do not linger.

> **Going deeper.** Secured lists complement, rather than replace, item-level access and burst fan-out. See Workspaces for workspace scope and Replicate output from your source system for value-driven fan-out.

Contribution processes

reference administrator

The **Contribution** tab of **Settings ▸ Performance** is the operations console for the contribution runtime. Its heading reads **Contribution worker processes and sessions**. Contribution input forms run in isolated, out-of-process worker processes; this console makes that fleet visible per server, governs the memory safety valve that admits or denies new contributor sessions, and packages contribution session logs for support.

Contribution memory scales with cells multiplied by concurrent sessions, so a busy fleet can saturate. This console is where that saturation is seen and relieved. Settings ▸ Performance is administrator-only, and so are its control verbs.

The console is live only. It stores nothing: every figure is the present moment, refreshed on a poll, and no history, time series, or earlier peak is retained. Capture anything worth keeping at the time it is seen. To measure a form, or to keep a record, use campaign profiling instead.

> **Looking for this under Auditing?** It used to be a **Contribution Processes** tab on **Settings ▸ Auditing**. It sits on **Settings ▸ Performance** now, beside the **Distribution** tab, because it is a live monitor rather than a record of what happened. Nothing about the console itself changed.

The console layout

The screen is master-detail: a left rail lists every online worker grouped by server, and a right pane inspects the worker you select.

Element	What it shows
Server group (left rail)	Each online server heads a group of its workers and shows a used-memory percentage, colored by pressure. An offline server shows a "last seen" note instead of workers.
Worker row	A two-segment memory bar (idle base vs. sessions alive), a flagged state when the worker is draining or at its session capacity, and the worker's active-over-maximum session count.
Worker inspector (right pane)	A stacked memory breakdown (idle baseline, sessions alive, headroom), a stat grid, and the worker's per-session table.
Auto-refresh · 10s	Repaints the telemetry every 10 seconds; a manual **Refresh** is also available.

The worker states are **Active**, **Draining**, **At capacity**, **Warm**, **Legacy 1:1**, **Idle**, and **Stopped**.

The inspector's stat grid carries four fields: **Workspace**, **Sessions** (active over maximum), **Served (lifetime)**, and **Uptime**. Its session table has six columns: **User**, **Opened** (a time, with no date), **Input form**, **Calc** (Auto or Manual), **Workbook** (Loaded or Unloaded), and **Actions**.

Headroom is measured against the busiest worker on the same server, not against the machine. It is the distance between this worker and that peer, so a large headroom figure does not mean the server has free memory. For the same reason the memory bars are scaled per server and are not comparable across servers.

Control verbs

The console can act on the live fleet. These are **administrator-only**, disruptive, and available only when a worker is controllable. Each one is recorded in the System audit log.

Verb	Effect
Force GC	Runs garbage collection on the worker to reclaim memory.
Drain	Stops the worker taking new sessions so it can wind down.
Shutdown worker	Stops the worker process.
Force-unload workbook	Releases a session's loaded workbook. Disabled when no workbook is loaded.
Kill session	Ends a live session. It confirms first: "End contribution session '{id}'? The user's unsaved edits are recorded and the form lock is released."

Killing a session reclaims a stuck session without losing the contributor's work in progress and without leaving the form locked.

The 90% memory admission valve

The same per-server memory reading drives a hard safety valve on the session-launch path. When a contributor opens a form, Reportworq checks the memory of every online server (local and load-balanced) and steers the new session to the online server with the most available memory that can still accept one.

A server can accept a new session only while its memory used is below 90%. If no server qualifies, meaning every online server is at or above 90% memory used, the launch is denied and the contributor is told:

> Unable to launch new Contribution session because all servers are at capacity. Please try again later.

Below the threshold, the load balancer steers the session to the least-loaded server. Watching per-server memory climb toward this valve during a live round is the signal to drain or shut down a hot worker, or kill a runaway session, to free headroom before contributors are turned away.

Session logs, for support

A toolbar handles contribution session logs across the whole fleet. Logs from remote servers are pulled on demand before packaging.

Action	Effect
Download logs	Gathers logs from all servers and saves a `Reportworq_Contribution_Session_Logs_<timestamp>.zip` locally.
Send to Support	Refreshes the remote logs, uploads them to the Reportworq support system, and notifies staff.
Clear logs	Clears all contribution session logs. Destructive; confirmation required.

When to use it

- **Diagnose contribution memory pressure.** During a large collection round, watch per-server memory, spot the hot worker, and drain or shut it down (or kill a runaway session) before contributors hit the admission valve.
- **Escalate to support.** Reproduce a contributor-reported problem, then Send session logs to support (or download the zip) so staff can investigate the exact worker and session.
- **Plan capacity.** Before opening a campaign, read per-server memory and worker counts to judge whether the fleet has headroom.

Notes and limits

- The console operates on live worker processes and sessions; the control verbs (drain, shutdown, kill) are disruptive, administrator-only, and gated to controllable workers.
- It does not profile or change a form's content. Measuring a single form's footprint offline is the job of the form memory profiler, and form content is edited in the campaign editor.
- A contributor never sees this surface, but is directly affected by the 90% valve.

> **Going deeper.** For the sibling **Distribution** tab on the same screen, which ranks report runs across releases and nodes, see Compare performance across reports. For the write-back processing this fleet's collected input feeds into, see Data collection processing. For the wider multi-server picture, see Deployment topology.

Migrate to a new server

how-to administrator

Migration moves an existing Reportworq instance to different hardware while keeping its full state: jobs, schedules, contacts, datasources, settings, and stored credentials. Because that state lives in the **Repository folder**, a migration is fundamentally: install the same product on the new server, then replace its fresh Repository with the old one, done in the right order, with the service stopped at the swap.

The procedure below is written for a **server-to-server move on Windows or Linux**, which is the same shape on both: install, stop the service, swap the Repository, start. Only the paths and the service manager differ, and both are called out where they matter.

Container deployments do not migrate this way at all. A Docker or Azure App Service deployment keeps its state on a mounted repository, so you move it by pointing a new deployment at that same repository. There is no folder to swap. See Deployment topology.

Before you begin

- A new server that meets the system requirements, on either Windows or Linux.
- Administrator rights on both servers.
- Your license key, and connectivity to the licensing service (or the manual activation path). Optionally, a non-production license key if you want to test the move before cutting over, see Move the license.
- A maintenance window.

Tip: before you migrate, run the repository metadata report on the old instance. It exports the whole distribution configuration to a spreadsheet, giving you an inventory to check the migrated instance against.

Where the Repository lives

The Repository is the single folder that holds the instance's state. On Windows it sits under the install folder at `C:\Program Files\Reportworq 6\Repository` by default; on Linux it sits under the install root alongside `settings.json`. If it was relocated (Settings > Configuration > Web Server, Repository path), copy it from wherever it actually lives, not the default. Everything except the license and locally-stored report templates travels inside this folder.

Migration procedure

Follow these steps in order. The order matters, swapping the Repository while the service is running risks corrupting it.

1. **Copy the Repository folder** from the old server (default `C:\Program Files \Reportworq 6\Repository`) to safe storage. This carries all instance state.
2. **Document the service identity** on the old server: the Bootstrap Service logon account on Windows (Local System or a specific domain account), or the user the systemd unit runs as on Linux. You must reproduce it on the new server or network access will break.
3. **Install Reportworq on the new server** using the same installer, see Install Reportworq. This creates a fresh, empty Repository.
4. **Stop Reportworq** on the new server. On Windows that is the **Reportworq 6 Bootstrap Service** (`services.msc`); on Linux, stop the systemd unit.
5. **Reproduce the service identity** on the new server and its file permissions, see Install Reportworq. On Windows, match the Bootstrap Service logon account if it was a domain account rather than Local System; on Linux, match the user the systemd unit runs as.
6. **Replace the Repository.** Delete the freshly generated Repository folder on the new server and put the copied old Repository folder in its place.
7. **Start the service.**
8. **Sign in and validate.** Check jobs, schedules, datasource connections, and run a test job.

Warning: always stop the service, then swap, then start. Replacing the Repository folder while the Bootstrap Service is running risks file conflicts and store corruption, because the running instance holds locks and may write over the swap.

Move the license

The license is **not** part of the Repository swap.

Standard move

1. On the **old** server, open **Settings > Configuration > License** and select **Release** to return the seat.
2. On the **new** server, **Activate** with the same key (or use **Manual activation** if the server is air-gapped).

Warning: migrating without releasing first can strand the seat. See Licensing and entitlements.

Test with a non-production license key first

If you want to validate the new server while the old one keeps running, request a **non-production license key** from Reportworq support and activate the new server with it. This lets you prove the migrated instance, its jobs, schedules, and connections, without taking the production instance offline. When you are ready to sunset the old server:

1. Release the non-production key on the new server.
2. Release the production license on the old server.
3. Activate the new server with the production key.

Things that do not travel in the Repository

- **Locally-stored report templates.** Reports served from a local file path on the old server (a source report provider pointing at a local folder) are **not** inside the Repository. Copy them to the new server separately, or re-point the provider, or those reports are lost. This is the same rule that governs installation: sources must be reachable by the service account.
- **The service logon account.** If the old instance ran jobs under a domain account with rights to network shares, the new instance must use the same identity (or an equivalent), otherwise jobs that read or write network paths fail after the move even though the Repository is intact.

What is portable

Encrypted credentials travel with the Repository. Passwords and connection strings are encrypted with a portable static key (not tied to the Windows machine), so the ciphertext in the migrated Repository decrypts on the new server with no re-entry.

Move a subset of content instead

To move individual jobs or folders between instances rather than the whole instance, use content-level Export/Import ("Send to") instead of a Repository swap:

1. In the source instance, open the workspace and select the job or folder to move.
2. Use **Send to** (or Export) to package the content, or send it directly to the destination workspace or installation.
3. In the target instance, import the content into the destination workspace.
4. Reconnect any data sources or providers the imported content depends on, then run a test job to confirm it works on the new instance.

Use this for a partial move; use the full procedure above to move the whole instance. For the detailed content-move steps, see Workspaces.

Decommission the old server

Once the new server is validated and the license has moved, retire the old instance:

1. Release the old server's license if you have not already (Settings > Configuration > License, **Release**).
2. Uninstall Reportworq from Windows **Add/Remove Programs > Uninstall**, see Install Reportworq.

Warning: some files remain after uninstall. Do not delete the residual files if there is any chance you will need to migrate again, they hold Repository and configuration state.

Notes and limits

- Migration as described here is a server-to-server move, on Windows or Linux. Container deployments are relocated instead by re-pointing a new deployment at the mounted repository, see Deployment topology.

> **Going deeper.** The backup-and-restore-into-Repository path is the recovery-style variant of this move, see Update Reportworq and Server configuration.