

First Workflow: from design to result

A small exercise covering Mapping, Session, Workflow and Monitor.

Before you start

Prepare a laboratory location in the selected tenant and permission to create, publish and run artifacts. You need a worker compatible with the files and an output you can inspect. The goal is retaining active customers, not load testing. Use synthetic data only.

The three artifacts serve different purposes: Mapping transforms; Session pins the revision and execution context; Workflow coordinates the task. Review each publication before moving to the next level.

Procedure

1. Create a UTF-8 CSV with header `id,name,status` and three synthetic rows. Include two `active` rows and one `inactive` row.
2. In Mappings, connect Flat File input → Row filter → Field selector → Flat File output.
3. Set Row filter to `field=status`, `operator>equals`, `value=active`. Set Field selector to `fields=["id","name"]`. Enable the output header.
4. Validate, save the draft and publish the Mapping. Create a Session referencing that revision, configure worker and file bindings, and publish the Session.
5. Create Start → Session → End in Workflows. Select the published Session revision, validate, save and publish.
6. Run manually and open the attempt in Monitor.

Synthetic example

Input:

```
id,name,status
1,Jorge,active
2,Ana,inactive
3,Bea,active
```

Copy

Expected output with these settings:

```
id,name
1,Jorge
3,Bea
```

Copy

Names and identifiers are fictitious. The file must be available to the chosen runtime; writing its name in a description does not create a binding.

Verify the result

Confirm terminal Workflow and task states, two data rows, header `id, name` and no `status` column. Verify that the Session references the revision you published. A file from an earlier attempt is not evidence of the current result; use a unique output or a mechanism identifying the attempt.

Only after acceptance should you consider scheduling. Review timezone, next execution and permissions. An active schedule does not replace verifying the next actual attempt.

Troubleshooting

For no rows, compare `active` exactly with source values: `equals` is case-sensitive. If `status` appears, inspect Field selector and the consumed revision. For waiting, inspect worker capacity and policy. For missing files, correct Session bindings. Do not blindly repeat writes after a timeout: check Monitor. DovaAI explains which components to use and how to configure them; you create, edit, publish and run the artifacts.

Environment context

Before using this module, confirm Tenant, Domain and Environment in the user menu. The initial environment is Development. Module, Domain and environment capabilities apply together; Production is read only without explicit authorization. Switching the selector does not move artifacts or publish or execute their data. See [environments and recovery](#) and [environment promotion](#).

Normalize customers with verifiable rules

Clean text, replace an exact value and limit output columns.

Before you start

Define the rule first: trim `name`, change exactly `jorge` to `Jorge`, and retain only `name` and `amount`. This is not general capitalization. Use a laboratory CSV with headers, UTF-8 encoding and a separate destination. Prepare permissions and compatible runtime before testing.

No SQL expression is needed for this literal change. For general first-letter capitalization, Clean text offers `capitalize`; for each word, use `title`. This example deliberately keeps the exact Replace values rule. Calculated field interpolates text without executing `LEFT` or formulas.

Procedure

1. Add Flat File input with `format=csv`, `delimiter=`, and `hasHeader=true`. Review field names and header.
2. Add Clean text with `fields=["name"]` and `operations=["trim"]`. This allows a value containing surrounding spaces to match the following exact rule.
3. Connect Replace values with `field=name`, `find=jorge`, `replacement=Jorge`, `matchMode=exact`, `caseSensitive=true`.
4. Add Field selector with `fields=["name", "amount"]`. This removes all other columns from schema and rows.
5. Connect Flat File output with `includeHeader=true` and a test filename. Validate, save and publish; prepare the Session and Workflow for execution.

Synthetic example

Input:

```
name,amount,internal_note
jorge ,12.5,sample-a
ana,8,sample-b
```

Copy

Expected output:

```
name,amount
Jorge,12.5
ana,8
```

Copy

Order is intentional: Trim before Replace values. The rule does not change `ana` or convert amounts; for typed numbers, add Convert type and define its error policy.

Verify the result

Check two rows, header containing only `name, amount` , Jorge replacement and Ana preservation. The `internal_note` column must not appear. CSV output writes the received schema; its filename does not select fields.

To check required values, you can add Required fields with `fields=["name"]` and `maxInvalidPercent=0` . It fails the entire execution for an invalid row; it neither fills values nor drops records. Do not add checks without deciding how their failures should be handled.

Troubleshooting

If replacement fails, inspect whitespace, case, field name and `matchMode` . For extra columns, inspect the saved Mapping and bound revision; do not assume Field selector retains unselected fields. If general capitalization is required, specify it as a different requirement; this replacement is not a universal solution. Before repeating a real load, verify idempotence and destinations. A correct example does not establish financial precision or rounding policy. Record the accepted rule before changing the production design.

Regional summary with Join and Aggregate

Join two synthetic sources and verify a total by region.

Before you start

This exercise uses two sources: transactions and a customer catalogue. The result must sum only approved transactions by region. Prepare runtime supporting two inputs and Join; generic linear streaming does not execute this graph. Both files must be available through authorized bindings.

Use the small tables in this chapter without downloading old fixture packages. Before designing, verify that customer `customer_id` is unique: duplicates can multiply joined rows and inflate totals.

Procedure

1. Create a pipe-delimited TXT containing `customer_id`, `amount`, `status`. Set Flat File input to `format=txt`, `delimiter=|`, `hasHeader=true`.
2. Create Excel with a Customers sheet and `customer_id,region` header. Select that sheet in Excel input, header row 1 and data starting at row 2.
3. Convert `customer_id` to `integer` on both branches. On transactions convert `amount` to `number` with `outputField=amount_number` and `onError=fail`.
4. Clean customer `region` using `trim` and `upper`. Filter transactions to `status=APPROVED` with Row filter.
5. Join both branches with `inner`: transactions left, customers right; keys `customer_id`, `keyType=integer`, `nullsEqual=false`. Select the actual identifiers of connected components.
6. Group by `region`: `metric=sum`, `field=amount_number`, `outputField=total`. Sort `total` descending and select `region,total` before CSV output.

Synthetic example

Transactions:

```
customer_id|amount|status
1|10|APPROVED
1|5|APPROVED
2|7|APPROVED
2|99|REJECTED
```

Copy

Customers sheet rows:

```
customer_id,region
1,NORTH
2,SOUTH
```

Copy

Expected result:

```
region,total
NORTH,15.0
```

Copy

Textual number representation can vary with numeric formatting; verify numeric totals 15 and 7 plus configured order.

Verify the result

Calculate the result independently: ignore REJECTED and sum 10+5 for NORTH and 7 for SOUTH. Expect two rows and exactly two columns. Confirm Join matched all approved identifiers without expansion caused by duplicates. Save and publish the accepted Mapping, configure both Session bindings and run a laboratory Workflow.

Group aggregate returns only grouping keys and metric output; it does not retain the detail of each original transaction.

Troubleshooting

For missing regions, check key types, unmatched data and Join type. For inflated totals, inspect catalogue duplicates. For separate regions caused by spaces or case, check Clean text before aggregation. A streaming error can indicate topology incompatibility rather than bad data. For real amounts, define precision, currency and rounding separately; this example does not establish an accounting policy.

Secure operation: from laboratory to production

Check identity, dependencies and results before enabling recurring execution.

Prepare a laboratory

Use synthetic data, a separate destination and an authorized identity. Confirm the selected tenant and active role before creating objects. A Repository location organizes items; it does not grant permissions. Use a read-only source Connection when the exercise does not require changing the source. Enter credentials only through the protected Connections mechanism, never in names, descriptions, example parameters or messages to DovaAI.

Build a verifiable chain

1. Create a small Mapping and configure its source, transformation and target. To test exact replacement, use Replace values with `field=name`, `find=jorge`, `replacement=Jorge`, `matchMode=exact` and `caseSensitive=true`.
2. Validate and save the draft. Publish the accepted version: saving and publishing are separate actions.
3. Create a Session referencing that Mapping version. Review connections, inputs, outputs, execution profile and authorized capacity.
4. Create a Start → Session → End Workflow. Save, validate and publish deliberately.
5. Start one laboratory attempt; compare Monitor and the produced file with the expected result.

An execution file path must exist where the worker can use it. Attaching a file to DovaAI to inspect its schema does not establish a permanent worker input. Keep the laboratory destination separate from business output throughout the exercise.

Accept before scheduling

For this example, verify that `jorge` becomes `Jorge` and `ana` remains unchanged. If you add Field selector with `fields=["name", "amount"]`, check that the output schema and final header contain only those fields. A succeeded status does not replace this functional comparison.

Enable a schedule only after accepting the result. Check its timezone, published dependencies and destination effects. A public worker offers eligible capacity: it does not publish another tenant's Session or data. Available slots do not guarantee access to a Connection either. Record the accepted versions so a later change can be reviewed against the same baseline.

Resolve incidents without duplicate effects

If the client times out, inspect the attempt before submitting another. Cancel, terminate, rerun and recover are different actions. A retry may repeat external writes; review idempotency and persisted state. When a dependency changes, inspect consumed versions before publishing consumers again.

Keep authorized result evidence and remove only reviewed laboratory objects. Purge contents is destructive and requires scope confirmation; it is not a routine error-recovery step. For denied access, check membership, role, plan and resource eligibility before requesting additional permissions.

Environment context

Before using this module, confirm Tenant, Domain and Environment in the user menu. The initial environment is Development. Module, Domain and environment capabilities apply together; Production is read only without explicit authorization. Switching the selector does not move artifacts or publish or execute their data. See [environments and recovery](#) and [environment promotion](#).

CLI and API: controlled access and checks

Use the official distribution and distinguish a personal key from an MFA-protected web session.

Prepare the client

Download the distribution for your operating system from Developer Tools and compare its checksum with the published value. Treat that distribution's help as the reference for its commands. The CLI version can differ from the platform version; a package filename alone does not establish compatibility or integrity.

The CLI authenticates with a personal Dova API key. Create it through the authorized API Keys interface and keep its value in a private mechanism. This is not an OpenAI key. Avoid placing keys or passwords in command arguments, screenshots, shared files or histories. Interactive login prompts for the key without displaying it. Use an operating-system account appropriate for the work you are performing.

Read before changing anything

These commands contain no secrets and inspect the current context:

```
dova --version
dova --help
dova login
dova status --output json
dova repository list --output json
dova mapping list --output json
dova workflow list --output json
dova run list --output json
dova logout
```

Copy

`dova login` uses the public service configured by the distribution. Verify that status shows the intended tenant before executing actions with effects. The CLI saves authentication configuration locally; protect your operating-system account and log out when appropriate. Closing the terminal does not revoke a personal key.

Run an authorized test

The help for `dova workflow run --help` describes inputs, parameters and waiting. First select a published laboratory Workflow with accepted dependencies. The `--wait` option observes the attempt; a client timeout does not prove execution has ended. Check Monitor before resubmitting. The cancel, terminate, rerun and recover commands have different consequences and current-state checks.

Distinguish authentication flows

The web-session API retains login, MFA when required, a cookie and CSRF protection for changes. If login returns `authenticated:false` with `mfaRequired:true`, MFA is still incomplete; HTTP 200 does not mean authentication finished. After successful MFA, inspect the session and use its current CSRF value for the relevant web flow.

A CLI Bearer personal key does not become a web session and does not replace MFA for global DovaAI provider configuration. That setting belongs to the uniquely authorized master user through its protected flow. Do not copy browser cookies into commands.

Check results and errors

For 401, inspect the credential or session; for 403, inspect the role and tenant. A validation error requires correcting the indicated resource, not blindly repeating writes. Keep safe attempt identifiers for diagnosis without logging authentication headers. Verify an accepted response in the corresponding object or execution attempt.

Export an environment promotion for Git/DevOps

Configuration → Environment promotions can export the manifest of an applied release. The GitOps kit adds `promotion-export --release-id <id> --output dova/promotion-release.json` through `app/bin/dova_gitops.py`, using its authorized configuration and authentication. This command reads the release, verifies its digest and writes a local file; it does not promote, execute or modify Dova or Git.

Run it from the local repository prepared for the kit. Review the manifest before recording it in version control: it retains published artifacts, dependencies and provenance; secret references become placeholders. Recording repository, commit and tag does not authenticate a remote repository or push commits. GitHub remains a repository and GitHub Actions is not activated.

Environment context

Before using this module, confirm Tenant, Domain and Environment in the user menu. The initial environment is Development. Module, Domain and environment capabilities apply together; Production is read only without explicit authorization. Switching the selector does not move artifacts or publish or execute their data. See [environments and recovery](#) and [environment promotion](#).

GitOps 1.1.2: bind the source and target

Download the [GitOps kit 1.1.2](#) from an authorized Developer Tools session and compare the published SHA-256. Extract `dova_gitops.py`, `dova-gitops.example.json` and `README.md` into your local repository. Use Python 3.12 or later and personal keys through the protected variables named by `apiKeyEnv`; keep their values out of Git.

In `dova-gitops.json`, `source.environment` and `source.domainId` select the export source. `target.environment` selects the destination and `bindings.domainId` is its authoritative Domain. When present, `target.domainId` must match `bindings.domainId`. The tenant remains the one authorized by the personal key and must match the configuration; these fields neither change tenant nor grant access.

Omitted environment values use `development`; the default Domain is General (`dom_general`). Set the context explicitly for another Domain. Version 1.1.2 binds the Domain from `bindings` before the first query and retains the same Tenant, Domain and Environment for reads, draft reservations, saves, publications and edit-lease acquisition or release. Do not change the context of a client that has already sent requests.

Synthetic example: Finance in Test

This fragment extends the kit example. Replace the IDs with an existing authorized Domain and Folder; it does not create Finance or its resources:

```
{
  "source": {"environment": "development", "domainId": "dom_finance"},
  "target": {"environment": "test", "domainId": "dom_finance"},
  "bindings": {"domainId": "dom_finance", "folderId":
```

[Copy](#)

```
"fld_REPLACE_WITH_EXISTING_FINANCE_FOLDER"},
  "statePath": "dova/target-state-finance-test.json"
}
```

Also configure URL, tenant, key-variable names and any Connection, worker, profile or file bindings required by the bundle. From the local repository, validate and review before an authorized import:

```
python dova_gitops.py --config dova-gitops.json validate
python dova_gitops.py --config dova-gitops.json plan --plan-file dova/plan.json
python dova_gitops.py --config dova-gitops.json apply --plan-file dova/plan.json
```

Copy

`validate` checks the local bundle. `plan` reads the target without writing. `apply` creates or updates definitions and publishes the reviewed versions in Development, Test or Stage. Published source versions and their environment labels remain unchanged. Direct Production authoring is rejected before requests with `production_authoring_requires_environment_promotion`; use Dova's governed environment promotion for that destination.

Preserve state and resolve a wrong context

Use a separate `statePath` per target. State binds the origin, tenant, Domain and Environment; reuse in another workspace is rejected before querying Dova. An empty older state can adopt the selected context. A populated earlier state without scope requires manual reconciliation of its IDs and published versions. Preserve state and the journal when a write outcome is uncertain: never delete them or change their context fields to force another import.

If a Finance import attempts to use General, check `source.domainId`, `target.environment` and `bindings.domainId`, install kit 1.1.2 and generate a new plan. Historical kits 1.1.0 and 1.1.1 retain their original bytes; 1.1.2 is the workspace-scoped distribution for Dova 24.

Overview: understand workspace state

Interpret indicators and choose the right module for investigation.

Overview summarizes objects, execution activity and operational health for the active tenant. Its cards help you decide where to investigate; they do not independently prove transformation correctness or completion of a commercial operation. Every observation belongs to the authorized context shown in the account menu.

Before you begin

Use an authorized account and a laboratory workspace. Check tenant and role before interpreting indicators. Do not switch to a customer workspace merely to populate summary cards.

Procedure

1. Open the account menu and confirm identity, workspace and active role.
1. Open Overview and review object summaries. Follow a card to its inventory to check names, filters and states.
1. Inspect recent activity and identify the relevant attempt by identifier and time. Continue in Monitor for task details and safe messages.
1. Use Workers or Configuration when capacity or runtime indicators need diagnosis. Check compatibility and availability before changing policies.
1. After an authorized change, return to the summary and compare it with the underlying module. Use Refresh when you need to confirm current state.

Verify the result

You can explain the scope of each indicator and trace it to a specific object or attempt. Compare a synthetic Workflow with its latest attempt in Monitor. Preserve the distinction between a saved definition, a published version and an execution.

Errors and recovery

A zero count can mean an empty workspace or restricted scope. A loading error is not zero. If information appears missing, inspect filters, permissions and tenant. Do not repeat Save or Run solely because a card has not updated; inspect the existing operation first.

Synthetic example

In Training, find `w_CustomerNames` from Overview, open its definition and compare its latest attempt identifier with Monitor. This is a read-only exercise and does not require another execution.

Verification scope

Use synthetic data and the role authorized for the task. Compare a visual confirmation with the saved record or relevant attempt. Retain only safe identifiers and messages for support; do not share secrets, personal information or tenant data. If an operation outcome is uncertain, check its state before repeating it.

Environment context

Before using this module, confirm Tenant, Domain and Environment in the user menu. The initial environment is Development. Module, Domain and environment capabilities apply together; Production is read only without explicit authorization. Switching the selector does not move artifacts or publish or execute their data. See [environments and recovery](#) and [environment promotion](#).

Worker availability and view loading

A reference to a removed worker or a worker with an invalid identity does not prevent you from opening Overview, Monitor or Connections. That reference is not available for execution; valid workers remain visible within your permissions. A real storage or policy failure is reported as an error and is not converted into a zero counter.

Repository: organization and content purge

Organize Domains and Folders and remove content through explicit review.

Repository organizes Mappings, Sessions, Workflows and Connections. A Domain is a top-level grouping; a Folder belongs to one Domain. Location alone does not grant permissions. Purge contents and Delete are different operations: the former clears the reviewed scope while preserving the selected location; the latter removes a location only when blockers are resolved.

Before you begin

Confirm tenant and location-management permissions. Review dependencies and open editors before deletion. Use a synthetic structure and preserve any information you still need outside the scope being purged.

Procedure

1. Select the correct Domain. Create a recognizable Folder and check its path before saving objects.
1. Use the available All, Mappings, Sessions, Workflows, Connections or Shared views to identify content by type and identifier.
1. To clear a location, open Purge contents from its menu. Review the preview and blockers: its scope can include Mappings, Sessions, Workflows, drafts, Connections, saved versions and child folders.
1. Type the requested name exactly and select the acknowledgment. Submit once and wait for the verified result.
1. If the response is uncertain, use Check result. Do not repeat the purge until the earlier result is resolved. A stale preview requires Refresh and fresh confirmation.
1. If the location must also be removed, use Delete separately. A Domain containing Folders or a location containing artifacts remains blocked.

Verify the result

Check that only reviewed content was removed. The selected location and execution history are preserved. If you subsequently removed an empty location, confirm its absence from the tree. A filtered grid without rows does not prove there are no dependencies.

Errors and recovery

Protected locations or inactive folders have restrictions. Resolve a blocker through supported actions rather than editing stored records. Distinguish a loading failure from an empty inventory. Purge does not promise undo or automatically delete its target Domain.

Synthetic example

Create Training/CustomerDemo and save m_CustomerNames there. In a disposable laboratory, review Purge contents for CustomerDemo and cancel before confirmation if you are only learning.

Verification scope

Use synthetic data and the role authorized for the task. Compare a visual confirmation with the saved record or relevant attempt. Retain only safe identifiers and messages for support; do not share secrets, personal information or tenant data. If an operation outcome is uncertain, check its state before repeating it.

Environment context

Before using this module, confirm Tenant, Domain and Environment in the user menu. The initial environment is Development. Module, Domain and environment capabilities apply together; Production is read only without explicit authorization. Switching the selector does not move artifacts or publish or execute their data. See [environments and recovery](#) and [environment promotion](#).

Recover a failed load

If context or inventory cannot load, Repository shows "Repository could not be loaded. Try again." and a Retry action to query the same authorized context again. Check the connection and selected Tenant, Domain and Environment. A load failure does not mean the repository is empty or that a Trial has lost artifacts. Do not recreate locations or delete drafts to resolve a read error.

Expired temporary drafts are excluded from inventory without turning the context query into a write inside its snapshot. Saved artifacts and existing publications are preserved. If the problem continues, give support the time and HTTP status, without credentials or tenant data.

Retry retains the failed query's Tenant, Domain and Environment; it does not switch environment or substitute another inventory. If you lack workspace access, open an authorized workspace from the user menu before continuing.

Mappings: design and publish transformations

Build the graph, configure components and publish a reviewed version.

Before you start

A Mapping defines the transformation algorithm. A Session supplies execution context and a Workflow coordinates tasks. Select the correct tenant and an authorized Repository location. Editing permission is required; publishing and running are additional actions. Prepare synthetic input and expected output without credentials or customer data.

A component in the palette does not guarantee compatibility with every worker. Check the runtime, especially for external connections and graphs containing Join.

Procedure

1. Open Mappings and create a draft in the appropriate Domain and Folder. Use a descriptive name.
2. Add components from their categories and connect each output to the next input. Select a component to configure its inspector.
3. Set source headers, names and types before configuring transformations. Review Join keys and the columns reaching the target.
4. Use Validate to resolve graph problems. Save persists the draft. A local-save indicator does not replace this action.
5. Publish only after saving every change. Publish requires a saved draft with no pending edits; it does not execute the Mapping.

Field selector: keep only selected fields

To retain only two CSV columns, connect Flat File input → Field selector → Flat File output. Set `hasHeader=true`, `delimiter=`, and `fields=["name", "amount"]`. The selector preserves values and rows but removes all other columns from the schema. The target uses that schema; `fileName` does not select fields.

Verify the result

Reopen the draft to verify persistence. Review the published revision before binding it in a Session. Execute a sample using authorized capacity and compare headers, rows and values with the expected result. Successful structural validation does not establish that files exist on a worker or that the output meets the business rule.

Ask DovaAI which components, connections and values fit your goal. The assistant gives brief instructions; you configure, save, publish and run through the normal Dova controls.

Troubleshooting

If Publish is disabled, check saving, pending changes, location, permissions and editing locks. If extra columns appear, inspect saved Field selector settings, the connected branch and the revision used by the Session. Correct runtime bindings for missing files. For streaming incompatibility, check the adapter: generic streaming requires a supported linear path. Do not remove necessary steps merely to make validation pass.

Apply different operations to different fields

Connect two Clean text components in series:

Raw file → Clean text 1 → Clean text 2 → Target

1. Clean text 1: Fields = `first_name` ; Operations = `trim` .
2. Clean text 2: Fields = `last_name` ; Operations = `upper` .

Both fields travel together. Each component changes only its selected field and preserves the others. These two operations need neither a split flow nor Field mapper.

Delete a canvas selection

Select the item and press Delete with focus inside the canvas. The shortcut must not remove components while typing in an input, textarea or another editor. Review affected connections and validate the graph after deletion; the change remains in the draft until you save and publish.

Columns for dependencies and publication

In Columns, choose Used by (Sessions), Location, Last published and Description. Used by (Sessions) counts the Sessions using the Mapping; select the number to inspect their names and referenced Mapping versions. Location combines Domain and Folder. Last published shows the most recent publication time and publisher separately from Updated, which can reflect draft changes. Description summarizes the Mapping purpose. When historical evidence is missing, the grid reports that instead of inventing a time or user.

A new grid configuration displays Mapping, Version, Status, Used by (Sessions), Location and Updated. Existing column choices are retained; enable additional columns in Columns. The Status filter offers Draft and Published, matching the editing and publication lifecycle.

Full screen without losing your work

Full Screen expands the Mappings or Workflows canvas in the same editor. Entering and leaving updates the layout immediately and preserves selection, zoom and position. Use the same button or Escape to return. DovaAI remains available in this view; expanding the canvas does not save or publish the draft.

Grid actions

When an Actions option starts, a progress indicator appears on its control and blocks other actions in that grid while the request is pending. Wait for the result before repeating it. A started request does not establish that saving, publication or another operation has finished.

Environment context

Before using this module, confirm Tenant, Domain and Environment in the user menu. The initial environment is Development. Module, Domain and environment capabilities apply together; Production is read only without explicit authorization. Switching the selector does not move artifacts or publish or execute their data. See [environments and recovery](#) and [environment promotion](#).

Compact list and icon buttons

The list brings Search, Status and actions into one compact toolbar. Refresh, Columns, Export CSV, Delete selected, New Mapping and Clear filters use icons; hover over a button or focus it with Tab to identify its action. Actions still respect selection, permissions and disabled states.

The Mappings list has no Environment or Runtime filter. Its environment comes from the active context in the bottom operational status bar; the worker and capacity are configured in the Session. Use Search, Status and column filters to inspect objects in that context.

Design, save and publish in the editor

Advanced, Validate, Save draft and Publish appear as icon buttons with accessible names. Button help identifies the action and, where applicable, the save or publication environment. The header does not repeat a DEV badge: confirm Tenant, Domain and Environment in the bottom operational status bar.

Mappings has no Run button. A Mapping defines the transformation; to execute it, save and publish the reviewed version, select it in a Session and configure its inputs, targets and compatible worker. Run the Session directly or through an authorized Workflow. Publishing a Mapping neither starts an execution nor assigns capacity.

Component Configuration

Select a component to inspect its sections and fields in Configuration. Labels, controls and help fit the available width. For sources with Header mapping, the sample-load, add-field and delete buttons retain their own space beside the title. Review Source, Output and Type in the table; scroll it horizontally when the columns do not fit the panel. A header sample configures the schema and does not replace the file the Session reads on its worker.

Sessions: bind a Mapping to runtime

Configure versions, parameters, connections and files for Mapping execution.

Before you start

A Session reuses a published Mapping in an execution context. It does not redefine the algorithm. Prepare an accepted revision, authorized location and compatible capacity. Keep editing, publishing and execution permissions separate. Browser-local files do not automatically appear in a worker filesystem.

Choose a laboratory environment and destination before testing writes. Classifying the environment as Production does not grant permission or publish the artifact.

Procedure

1. Create the Session and choose the published Mapping. Check its identifier and revision, not only its name.
2. Review the environment and worker policy. Keep inherited selection unless an explicit assignment is needed; any choice must satisfy capabilities and permissions.
3. Complete connections and input/output bindings. Use paths accessible by the selected worker and an exclusive test destination.
4. Configure declared parameters with correct types. Task overrides belong to the Workflow, not Session layers.
5. Review timeout, retries and recovery. Save the draft and publish the accepted revision before consuming it in a Workflow.

Synthetic example

A Mapping reads CSV `name, amount` and retains both columns. Create two Sessions referencing the same Mapping: one for Test and another for Stage. Use equivalent inputs and different outputs. Path differences belong in bindings; there is no need to duplicate the algorithm merely to change a file.

Verify the result

Reopen the Session and verify Mapping reference, revision, environment, worker policy and bindings. For parameters, inspect the source of each resolved value. Create a laboratory Workflow for each context and compare output schema and data. Successful saving does not establish that connections or files will be accessible during execution.

When a Mapping changes, review the dependency notice and deliberately accept the new revision. Then check Workflows consuming that Session.

Troubleshooting

For a missing file, correct its binding and check worker permissions. For unavailable capacity, distinguish worker state, slots, requirements and tenant policies. Fix invalid JSON without discarding other draft settings. A checkpoint only helps when usable state exists; it does not automatically reverse external writes. Before repeating a load, determine whether the destination supports retries without duplicates. Record which published revision each test consumes.

Runtime: profile, filenames and variables

In Runtime select Execution profile and compare input, memory, disk and timeout limits with worker capacity. An automatically selected profile can allow fewer bytes than your file; free VM disk does not increase that limit or isolated executor temporary memory.

Available runtime variables shows the name, meaning, Example preview and Last resolved value. The example helps prepare configuration; the resolved value belongs to the identified run and date. Sensitive values are masked. When no compatible execution is available, an example does not substitute for an actual value.

For output names use Target filename timestamp and Target filename template. `%RUN_TIMESTAMP%` resolves when the attempt starts. Enabled activates it for that Session; Disabled disables it; Inherit follows the global preference. Writing the token into a template while inherited mode is disabled does not activate substitution. Filename preview · example illustrates a filename rather than proving a run's actual output. Save and publish so consumers can adopt your change.

General: choose location before Mapping

After Name, complete Domain → Folder → Mapping, followed by Environment and Status. Available mappings are filtered by Domain and sharing permissions and load when the form opens, without reselecting Domain. Folder organizes the Session and does not require its Mapping to be in the same folder. After changing location, review the choices and keep a valid reference before saving.

Optional TRACE

In Sessions → Runtime, TRACE offers Inherit from Workflow, Enabled or Disabled. Inherit from Workflow follows the invoking Workflow; a directly executed Session without a Workflow resolves inheritance as disabled. Enabled and Disabled override inheritance for that Session. Inherit from Workflow is the default, and new Workflows have TRACE disabled.

This setting controls TRACE detail generation for new attempts; it neither removes historical evidence nor hides INFO, WARNING or ERROR. Save, publish and update Workflow dependency versions when needed to execute the accepted setting.

Environment context

Before using this module, confirm Tenant, Domain and Environment in the user menu. The initial environment is Development. Module, Domain and environment capabilities apply together; Production is read only without explicit authorization. Switching the selector does not move artifacts or publish or execute their data. See [environments and recovery](#) and [environment promotion](#).

Unassigned worker

Before running again, select compatible authorized capacity and review the environment pool. Unassigned does not silently choose another worker. You can provision a new VM and assign it explicitly. Permissions, resource ownership, the protected base server and pending operations remain checked; removal does not bypass these protections.

List toolbar and environment scope

In the Sessions list, action buttons use icons with help and accessible names and align with Search and filters. Identify them by hovering or moving focus with Tab. The Environment filter is removed from this list: it uses the Tenant, Domain and Environment context selected in the user menu and visible in the bottom operational status bar. Runtime and the other filters remain available. This list change does not remove the Session Runtime configuration or its worker requirements.

Workflows: coordinate and execute tasks

Define dependencies, publish revisions and verify each attempt.

Before you start

A Workflow coordinates tasks and dependencies. Session tasks execute the context associated with a Mapping; Start, Decision, Timer and End express orchestration. Prepare a published Session, compatible capacity and a laboratory destination. Review editing, publishing, execution and scheduling permissions separately.

Start with a small sequence. Adding parallelism before verifying output makes it harder to distinguish data errors, dependency problems and worker availability.

Procedure

1. Create a Workflow in the authorized tenant and location. Add Start, Session and End.
2. Select the Session and check `sessionVersion`. Connect Start → Session → End.
3. Configure task parameters only when declared and permitted. For multiple parents, review `joinPolicy=any` or `all`; visual position alone does not define eligibility.
4. When a Decision is required, configure its True condition and review the complementary False branch. Timer uses `delaySeconds`, from 1 to 604800.
5. Validate, save and publish. Run manually before enabling a schedule. Saving and publishing do not start execution.

Synthetic example

Prepare a Session that filters `status=active` and writes a test CSV. The smallest Workflow places that Session between Start and End. Expect one Session execution and an output containing only active rows. To study waiting, add a 5-second Timer before the Session while keeping the same expected data result.

Verify the result

In Monitor, locate the attempt identifier and review its tasks. Confirm the consumed Session revision and associated Mapping. Compare output, schema and rows with the expected result. A skipped branch is not a failed task.

After manual acceptance, configure an authorized schedule, timezone and enabled state. Check the next planned execution and then an actual attempt. An enabled schedule expresses intent; it does not establish successful completion.

Troubleshooting

The validator accepts 2–100 nodes and 1–300 links and rejects cycles, repeated identifiers and invalid dependencies. For a waiting task, inspect parents, worker, capacity and policy. For failure, find the first relevant error. Before retrying, review whether writes can repeat without duplicates. When dependencies change, deliberately review and republish consumers; a new revision does not establish automatic acceptance by every reference. Keep the earlier attempt identifier for comparison.

Consumed and available versions

The compact dependency chain distinguishes Workflow, Session and Mapping and displays the pinned version beside a newer available version. Opening Session or Mapping inspects the artifact; it does not adopt the newer version. Update versions changes draft references and displays the expected changes. Then use Save draft → Publish. Reload status only refreshes availability. A Monitor execution chain retains the historical versions actually used.

To promote an accepted version use Configuration → Environment promotions. Review the exact bundle and next-environment bindings. Publishing a version, updating references, promoting between environments and executing are separate user operations.

Workflow TRACE and inheritance

In the Workflow editor → Workflow settings, choose TRACE = Enabled when additional diagnostic detail is needed; the default is Disabled. Sessions set to Inherit from Workflow follow that value. A Session explicitly set to Enabled or Disabled keeps its own choice. Enabling one Session does not enable TRACE for other Sessions in the Workflow.

An execution retains the setting from the versions it actually consumed. Editing the draft does not alter an active attempt or add TRACE retroactively to earlier logs. INFO, WARNING and ERROR remain available without enabling TRACE.

Environment context

Before using this module, confirm Tenant, Domain and Environment in the user menu. The initial environment is Development. Module, Domain and environment capabilities apply together; Production is read only without explicit authorization. Switching the selector does not move artifacts or publish or execute their data. See [environments and recovery](#) and [environment promotion](#).

Compact list toolbar

The Workflows list aligns Search, the remaining filters and icon actions. Buttons retain help, accessible names, permissions and progress states; use the pointer or Tab to identify them. Environment is no longer a filter in this list: confirm the active context in the bottom operational status bar. Runtime and the other filters are retained. Workflow editing and execution controls keep their roles; this list adjustment does not change its Session versions or the capacity required to run them.

Monitor: investigate and verify executions

Connect global state, tasks, capacity and actual results.

Before you start

Monitor presents execution-attempt evidence. A saved or published Workflow is a definition, not an attempt. Keep the run identifier, approximate time and involved revisions. Select the authorized tenant and review read permission plus permission for each operational action.

Do not share input rows, credentials, cookies or unfiltered logs in an incident. A useful report includes state, task and a safe message without sensitive content.

Procedure

1. Open Monitor and narrow results by name, state or time. Check identifier and date: one Workflow can have multiple attempts.
2. Distinguish active from terminal states. Waiting for capacity differs from failure; cancellation and success are also different outcomes.
3. Open attempt details and inspect tasks and dependencies. Find the first relevant error, not merely the last skipped task.
4. For waiting, inspect worker, slots and execution context through authorized surfaces. Do not launch a duplicate because the grid is slow to refresh.
5. Before cancelling, terminating, recovering or repeating, refresh state and review that action's confirmation.

Synthetic example

Use a laboratory Workflow with synthetic CSV input. Run once with a correct path and again with a nonexistent path in the same laboratory. Locate both attempts by identifier. In the failed one, identify the affected task and missing-file message. Correct the binding and start a new attempt; retain earlier evidence according to the applicable policy.

Verify the result

For the successful attempt, compare the produced file with expected headers, row count and values. Success records execution outcome but does not independently verify the business rule. For the failed attempt, confirm that the explanation identifies the first error and that the new attempt uses corrected configuration.

Keep history cleanup separate: it does not cancel active work or replace a retention policy. Review scope and confirmation before removing evidence.

Troubleshooting

An empty grid may reflect filters; it does not prove server failure. A client timeout does not prove cancellation: query the attempt again. Recovery depends on available compatible checkpoints and does not automatically reverse external writes. Repeating may create another output or duplicate a load. If state changes while a dialog is open, reload before confirming. For provider errors, record only the safe message and time.

Compact logs and live execution progress

The log presents Timestamp and Message columns in compact single-line rows. Expand a record to inspect its formatted, color-highlighted structured detail; collapse it again to save space. Filter with Search log messages, Level, Component and Time zone; inspect level counts, Errors only and Download log. Follow tracks new messages when you are at the bottom; Pause/Resume pauses presentation only and does not stop execution. Wrap fits long lines.

While details are visible, the view polls changes and retains previous data during delays or temporary failures. Cards use the latest reported counters. Progress can show phase, batches, processed/written rows, rate, duration and bytes when known. Staging, processing and writing are different phases: zero rows before reading do not establish inactivity. No percentage is invented without a reliable total.

The worker records safe causes such as an input exceeding the profile maximum or a protected existing destination, with available phase and limit details. Credentials and private provider responses remain excluded. For an old attempt without its original exception, a generic message remains incomplete evidence; do not assign an unrecorded cause.

List and Gantt

The list updates cells without replacing the whole table and preserves selection, filters and scroll. List and Gantt share filters. The compact Gantt distinguishes Workflow, Session and Mapping through kind and hierarchy; a Session appears under its Workflow. Open the specific attempt to inspect state, duration and versions. A Workflow bar spans orchestration; a Session bar represents that execution and must not be added again as extra Workflow duration.

Compact reading and stable updates

List aligns Execution, Duration and Status in compact rows. Purple WF and green SES badges, with the same icons as the workspace tabs, distinguish Workflow and Session in both List and Gantt; color indicators accompany the written state, and hierarchy can be expanded. Gantt text and tooltips follow Dova typography with lighter weights for labels and values.

Updates change existing rows and insert new executions matching the filters without visibly clearing or rebuilding the entire list. Selection, filters and scroll are retained. An execution outside the current filters does not need to appear.

TRACE is optional detail generated according to published Workflow and Session settings. Monitor's level filter chooses which existing evidence to display; it does not enable TRACE generation. Inspect Session inheritance when a Workflow and its tasks have different TRACE counts.

Environment context

Before using this module, confirm Tenant, Domain and Environment in the user menu. The initial environment is Development. Module, Domain and environment capabilities apply together; Production is read only without explicit authorization. Switching the selector does not move artifacts or publish or execute their data. See [environments and recovery](#) and [environment promotion](#).

Conversation and environment

DovaAI uses the authorized Tenant, Domain and Environment and selected artifact. Switching context cancels earlier requests and starts a conversation for the new environment. A pending reply from the earlier context is not added to the new conversation. The server verifies environment identity and versions; form values, records and credentials are excluded from context.

Worker availability and view loading

A reference to a removed worker or a worker with an invalid identity does not prevent you from opening Overview, Monitor or Connections. That reference is not available for execution; valid workers remain visible within your permissions. A real storage or policy failure is reported as an error and is not converted into a zero counter.

Connections: reusable access and credential custody

Configure metadata and authentication without exposing secrets.

A Connection separates endpoint metadata, connector, authentication method and protected credential custody from Mapping logic. Available fields and methods depend on the chosen connector. A technology appearing in the catalog does not prove that a worker has the driver, network path or permissions needed to use it.

Before you begin

Prepare an authorized laboratory resource, preferably read-only. Identify the worker performing the test and its runtime identity. Do not put passwords, keys or sensitive Vault references into descriptions or screenshots.

Procedure

1. Open Connections and create a record whose name describes its purpose. Select the connector and inspect required fields.
1. Enter authorized metadata: host or endpoint, port, database, schema or resource as applicable. Confirm values with the environment owner.
1. Choose an offered authentication method and custody mode. Passwordless uses a runtime identity; external custody requires a resolvable protected reference. Do not force hidden methods into the form.
1. Complete the protected input flow, review tenant and save. Check persisted metadata before testing.
1. Use Test for the saved Connection and read the safe result. Then run a small extraction through a Mapping/Session on a compatible worker.
1. Review consumers before rotating, revoking or deleting the Connection. Distinguish metadata editing from replacing credentials.

Verify the result

The Connection exposes only permitted metadata; its test reports the expected state and a small read produces known output. Compare identifiers with the editor. A successful connection test does not certify every write, retry or recovery mode.

Errors and recovery

For authentication errors, inspect method, custody, permissions and expiry. For network errors, check DNS, TLS and worker egress. A missing driver requires checking installed capabilities. Keep messages safe and never copy secrets into a description field for debugging.

Synthetic example

CustomerLabReadOnly addresses synthetic database training and table customers through a read-only identity. Select it in the appropriate source component and verify ten known records.

Verification scope

Use synthetic data and the role authorized for the task. Compare a visual confirmation with the saved record or relevant attempt. Retain only safe identifiers and messages for support; do not share secrets, personal information or tenant data. If an operation outcome is uncertain, check its state before repeating it.

Environment context

Before using this module, confirm Tenant, Domain and Environment in the user menu. The initial environment is Development. Module, Domain and environment capabilities apply together; Production is read only without explicit authorization. Switching the selector does not move artifacts or publish or execute their data. See [environments and recovery](#) and [environment promotion](#).

Prepare destination Connections

Before promoting to Production, prepare target-owned Connections with valid custody and permissions. An authorized administrator can register them or change metadata with `etl.connections.manage` and `etl.environments.production.edit`. This specific authorization does not unlock Production Mapping, Session or Workflow editors. Shared or ambiguous legacy Connections stay read only.

Promotion and Fork to Development previews offer only compatible existing destination Connections. Select each replacement explicitly; passwords are not copied and secret references are not fabricated. Connection Domain and Environment remain tied to their ownership context.

Worker availability and view loading

A reference to a removed worker or a worker with an invalid identity does not prevent you from opening Overview, Monitor or Connections. That reference is not available for execution; valid workers remain visible within your permissions. A real storage or policy failure is reported as an error and is not converted into a zero counter.

Notification channels: destinations and delivery

Separate channels, policies, templates and email transport.

A channel defines a reusable destination for authorized events. It does not itself determine when an event is generated or prove that a recipient received a message. Notification policy selects events and routing; templates define email content; SMTP supplies transport. These configurations require different checks.

Before you begin

Use a laboratory recipient or webhook you control and a role authorized to manage channels. Treat addresses and endpoints as sensitive where applicable. Do not use customer data to test an integration.

Procedure

1. Open Notification channels and check whether an existing authorized destination can be reused.
1. Create the channel with a recognizable label, offered type and required fields. Use protected input when credentials are needed.
1. Confirm tenant and destination, save and reopen the record to verify permitted metadata.
1. Use Test where available and send synthetic content. Check both the reported state and receipt in the laboratory system.
1. Associate events, severity and recipients through the relevant policy. Review suppression, deduplication, escalation, digest and quiet hours when configured.
1. For email, review the template and Email delivery separately. A content preview proves neither SMTP operation nor receipt.
1. Generate a controlled event and compare Delivery history with the receiving system. Restore temporary policies when finished.

Verify the result

The channel remains saved, its test reports a consistent result and the destination receives the expected message or suppression is explained. Keep only a safe identifier and timestamp. Local acceptance can describe handoff to transport rather than final receipt.

Errors and recovery

If a message does not arrive, first check whether the event was generated and whether policy suppressed it. Then inspect destination, authentication and transport through safe messages. Do not send repeated tests while an outcome remains pending. Review consumers before deleting the channel.

Synthetic example

Create LabOperations with an authorized test mailbox. Send a synthetic notification, locate its state in Delivery history and confirm subject, text and links in the mailbox.

Verification scope

Use synthetic data and the role authorized for the task. Compare a visual confirmation with the saved record or relevant attempt. Retain only safe identifiers and messages for support; do not share secrets, personal information or tenant data. If an operation outcome is uncertain, check its state before repeating it.

Workers: eligible capacity and privacy

Check health, compatibility and availability before assigning work.

Workers execute Sessions and Workflows under tenant policy. Heartbeat, VM power state and eligibility are different observations. A public worker can offer shared capacity; assigning a Session to it does not publish its definition, data or artifacts to other users or tenants.

Before you begin

Prepare an authorized host and know its owner. You need compatible capacity, permissions and network access to process resources. Input files must be available to the worker; selecting it does not copy them automatically.

Procedure

1. Open Workers and identify the record by identifier, tenant or pool and recent heartbeat. Do not rely on its name alone.
1. Open details and inspect version, resources, connectors and slots. Compare these with Session requirements and the Execution profile.
1. If registering a host, use authorized enrollment and keep the token within protected input. Confirm a valid heartbeat appears.
1. Distinguish restarting the worker service from powering on a provider VM. Provisioning and provider controls belong to Administrator.
1. Choose the default or a permitted explicit assignment. Check that the identity can resolve Connections and access the input file.
1. Run a synthetic Session and verify in Monitor which worker received the attempt and its result.
1. Before retiring or replacing capacity, inspect active jobs, references, default policies and dependencies.

Verify the result

The record is current, eligible for the task and completes a test with known output. Slots are weighted units: four slots can accept four weight-one jobs or two weight-two jobs, subject to other limits. Online does not mean unlimited capacity.

Errors and recovery

For missing heartbeat, inspect service and permitted connectivity. For ineligibility, inspect policies, capabilities, resources and lifecycle. Provider unavailable does not prove the VM is absent. Resolve failed removal through the supported procedure rather than deleting records to hide the problem.

Synthetic example

Assign s_CustomerNames to lab-worker with a synthetic CSV available there. Check the worker identifier and Mapping version used by the attempt.

Verification scope

Use synthetic data and the role authorized for the task. Compare a visual confirmation with the saved record or relevant attempt. Retain only safe identifiers and messages for support; do not share secrets, personal information or tenant data. If an operation outcome is uncertain, check its state before repeating it.

Environment context

Before using this module, confirm Tenant, Domain and Environment in the user menu. The initial environment is Development. Module, Domain and environment capabilities apply together; Production is read only without explicit authorization. Switching the selector does not move artifacts or publish or execute their data. See [environments and recovery](#) and [environment promotion](#).

Environment worker pool

In Configuration → Dova Integration Runtime, Environment worker pool identifies the current environment. Allowed workers assigns existing workers; Default worker must belong to that selection. Pool policy starts as Dedicated. Select Shared only when you explicitly want to share capacity and every participating pool also declares Shared. A matching name does not share a physical worker.

Save environment pool changes only the selected environment's policy and neither starts a job nor changes the tenant's global default. The review uses a configuration fingerprint; if another user changes it, Refresh pool lets you reload and review before saving. Controls require configuration and environment Runtime administration. Built-in is not offered as an execution worker.

Production does not automatically borrow the Development pool. New jobs require an explicit pool, apart from the public Development option for an eligible Trial described below. Legacy execution can retain an earlier worker only when the server verifies the exact published chain, configuration and permitted membership. Test and Stage consume the plan's Development bucket; Production uses the Production bucket. Pool membership does not replace capacity, online status, profile compatibility, quotas or concurrency.

Public Trial: platform-managed Development pool

For an eligible public Trial without an explicit Development policy, Dova derives a platform-managed public Development pool. It selects only public shared workers with verified references, authorized for the Trial and absent from other configured pools. It does not include private workers, create workers or save Runtime changes. The Trial does not gain permission to save pools.

An explicit Development policy, including an empty Allowed workers selection, takes precedence. This mechanism does not extend to Test, Stage or Production. Capacity, profile, isolation, quota and concurrency checks still apply; public access does not guarantee availability.

Before stopping or retiring capacity

Pause VM and Stop VM retain active-work, default-worker and shared-capacity checks; review references and impact before using them. Pause scheduling stops new assignments and lets current work finish. To retire the VM permanently, use Remove VM and unlink and review its confirmation: after verifying cloud absence, Dova clears assignments and shows Unassigned without deleting definitions or published versions. Assign compatible capacity explicitly before running again.

Actions require the module permission and the relevant environment's Runtime capability, or an explicitly granted global permission. A shared VM requires authorization across all its environments.

Removal and later assignment

Open Remove VM and unlink, review resource identity and impact on Sessions, executions and pools, then confirm the intended deletion. For a supported owned VM, manually dismantling every reference or first powering it off is not a prerequisite for this operation. Removing capacity can interrupt active executions; schedule the operation to suit your needs.

After confirming provider absence, Dova retires the active worker, clears Runtime references and shows Unassigned for Session bindings that used it. Definitions, published versions, configuration and audit history are preserved. If the VM is deleted directly in the provider, reconciliation applies the same treatment after verifying absence. A provider-read error is not evidence that the VM was deleted.

Before running again, select compatible authorized capacity and review the environment pool. Unassigned does not silently choose another worker. You can provision a new VM and assign it explicitly. Permissions, resource ownership, the protected base server and pending operations remain checked; removal does not bypass these protections.

Configuration: runtime, profiles and scheduling

Separate capacity policy, execution limits, calendars and health.

Configuration brings together Dova Integration Runtime, Execution profiles, Workflow Scheduler, Runtime health and Promotions. Policies select authorized capacity; profiles express limits; calendars schedule future orchestration; health describes current observations. Saving a policy does not establish that a worker is available or that an attempt completed successfully.

Before you begin

Confirm tenant and permissions before saving. Record the baseline so laboratory changes can be restored. Distinguish the default worker from a Session override and schedule timezone from Profile display timezone.

Procedure

1. Open Configuration and identify which of its areas addresses your need. Inspect before editing.
1. In Dova Integration Runtime, review the default worker and offered selection or affinity options. Check eligibility in Workers.
1. Compare a Session using Use tenant default with one using a permitted override. An override does not change the default for other Sessions.
1. In Execution profiles, inspect units and limits of the authorized profile. Change values only for a concrete operational reason and check inheritance.
1. In Workflow Scheduler, review calendar, timezone and enabled state. Associate it with the intended published Workflow after a successful manual test.
1. Save reviewed fields, wait for acceptance and reopen to confirm persistence. When nothing changed, Save can correctly remain disabled.
1. Inspect Runtime health separately and continue in Workers or Monitor to verify capacity and actual attempts.

Verify the result

Two synthetic Sessions, one using the default and one with a compatible override, show the expected worker in execution context. For a calendar, record timezone, next planned time and resulting run identifier; these are separate checks.

Errors and recovery

Do not increase limits to conceal an unavailable worker. Correct invalid units or ranges. Following a concurrent change, reload the baseline and compare before resubmitting. Disable temporary schedules when finished: closing Configuration does not disable them.

Synthetic example

Use Training with `s_Default` and `s_Override`. Keep transformations equal and change only the permitted worker selection; compare both attempts in Monitor.

Verification scope

Use synthetic data and the role authorized for the task. Compare a visual confirmation with the saved record or relevant attempt. Retain only safe identifiers and messages for support; do not share secrets, personal information or tenant data. If an operation outcome is uncertain, check its state before repeating it.

Environment promotions

Open Configuration → Environment promotions. Select a published Mapping, Session or Workflow and its source version. The permitted order is Development → Test → Production (Stage is optional when enabled). Each promotion keeps the exact Session and Mapping dependencies and creates separately published destination artifacts; source-environment originals remain intact.

1. Choose the next environment and define Connection replacements and Session settings: worker, profile, parameters, input and output bindings. Use existing IDs; credentials remain protected in Dova.
2. Complete Git provenance with an HTTPS repository and full commit SHA or tag when applicable.
3. Select Review promotion. Inspect dependencies, version changes, compatible workers, plan capabilities, available DPU and concurrency. Development, Test and Stage use Development processing; Production uses Production processing.
4. Apply only the reviewed preview and confirm. Production additionally requires approval permission and a comment. If destination state, definitions or plan change, review again.
5. Verify history and the resulting artifacts. Promotion does not execute the Workflow or enable its schedules; enable scheduling only after acceptance.

Export manifest downloads the immutable manifest and provenance for your repository or DevOps process. It contains neither credentials nor usable secret references. Dova does not push commits or activate GitHub Actions.

Restore this release restores historical definitions and environment settings through new destination versions, subject to authorization, confirmation and current checks. It does not undo written files, loads or external data changes. Revision and idempotency checks protect against duplicate application; after a conflict, refresh and compare before confirming again.

Environment context

Before using this module, confirm Tenant, Domain and Environment in the user menu. The initial environment is Development. Module, Domain and environment capabilities apply together; Production is read only without explicit authorization. Switching the selector does not move artifacts or publish or execute their data. See [environments and recovery](#) and [environment promotion](#).

Environment worker pool

In Configuration → Dova Integration Runtime, Environment worker pool identifies the current environment. Allowed workers assigns existing workers; Default worker must belong to that selection. Pool policy starts as Dedicated. Select Shared only when you explicitly want to share capacity and every participating pool also declares Shared. A matching name does not share a physical worker.

Save environment pool changes only the selected environment's policy and neither starts a job nor changes the tenant's global default. The review uses a configuration fingerprint; if another user changes it, Refresh pool lets you reload and review before saving. Controls require configuration and environment Runtime administration. Built-in is not offered as an execution worker.

Without an explicit pool, Production does not borrow Development capacity and new jobs do not run through an implicit default. Legacy execution can retain an earlier worker only when the server verifies the exact published chain, configuration and permitted membership. Test and Stage consume the plan's Development bucket; Production uses the Production bucket. Pool membership does not replace capacity, online status, profile compatibility, quotas or concurrency.

Colors per Tenant and Environment

Color identifies the active context in the bottom operational status bar summary, environment badge and Environment select border. Defaults are blue for Development, amber for Test, violet for Stage and red for Production. The environment name or code stays visible; color does not grant access or change your personal palette.

To customize it, first choose Tenant and Environment in the user menu, then open Configuration → Miscellaneous → Environment colors. Choose a color or enter a six-digit hexadecimal value, review the preview and use Save color. The setting is shared for that Tenant+Environment, including across Domains. Restore default prepares the default value and Save color applies it. Refresh loads the current configuration after a concurrent change. Configuration management permission is required; there is no separate per-Domain color or additional Tenant-wide base-color selector.

Pool profiles and limits

In Configuration → Dova Integration Runtime, the environment pool lets you choose Allowed workers, Default worker and Dedicated or Shared policy. A worker can belong to multiple environments only when all participating pools declare Shared. Available options also restrict execution profiles and default profile, concurrent slots, timeout and attempts. Review and save the selected environment's policy; these limits do not replace capacity, heartbeat, compatibility or plan quotas.

Plans and subscriptions: effective offers and continuity

Understand current terms without confusing offers, orders and subscriptions.

Plans and subscriptions displays effective commercial choices for the active tenant. Availability depends on tenant type, lifecycle, plan and authorized offers. A displayed price, a checkout order and an active subscription are different stages. Always read current amounts, currency, period and conditions in the platform; this manual does not establish prices.

Before you begin

Confirm tenant and role before opening checkout. Learn in a separately authorized payment test environment. A public trial should continue as the same operational tenant after accepted conversion; do not recreate the workspace to simulate conversion.

Procedure

1. Open Plans and subscriptions and inspect plan, lifecycle and capabilities available to your role.
1. Read the displayed offer version, price, currency, period, quantity, taxes and terms. Tenant-specific offers reference the versioned global catalog.
1. When evaluating an authorized purchase, review customer details, document, legal name, email, country, address and payment method before proceeding.
1. For an exercise, use only fixtures or an independently authorized sandbox. Never submit synthetic tokens to a live provider.
1. After an authorized purchase, inspect the definitive order and subscription outcome. Investigate the existing order before repeating checkout.
1. Confirm that tenant identity and Repository object identifiers remain unchanged and that capabilities match the accepted subscription.
1. For renewal or changes, inspect validity and contractual version again. A new price does not rewrite historical orders.

Verify the result

Conversion has an accepted commercial outcome, retains the operational tenant and grants matching capabilities. DovaAI allowances and user/capacity limits depend on the effective contract; do not infer unlimited access from an administrative title.

Errors and recovery

If an offer is missing, check eligibility, lifecycle, role and plan. If amounts disagree, confirm currency and quantity before submitting. A pending outcome requires resolving the earlier order rather than creating another as a test. Do not mark an invoice paid merely to enable features.

Synthetic example

In an authorized sandbox, review the Training offer, cancel before payment when only learning the form and verify that no unintended subscription was created.

Verification scope

Use synthetic data and the role authorized for the task. Compare a visual confirmation with the saved record or relevant attempt. Retain only safe identifiers and messages for support; do not share secrets, personal information or tenant data. If an operation outcome is uncertain, check its state before repeating it.

Environment context

Before using this module, confirm Tenant, Domain and Environment in the user menu. The initial environment is Development. Module, Domain and environment capabilities apply together; Production is read only without explicit authorization. Switching the selector does not move artifacts or publish or execute their data. See [environments and recovery](#) and [environment promotion](#).

Environment worker pool

In Configuration → Dova Integration Runtime, Environment worker pool identifies the current environment. Allowed workers assigns existing workers; Default worker must belong to that selection. Pool policy starts as Dedicated. Select Shared only when you explicitly want to share capacity and every participating pool also declares Shared. A matching name does not share a physical worker.

Save environment pool changes only the selected environment's policy and neither starts a job nor changes the tenant's global default. The review uses a configuration fingerprint; if another user changes it, Refresh pool lets you reload and review before saving. Controls require configuration and environment Runtime administration. Built-in is not offered as an execution worker.

Production does not automatically borrow the Development pool. New jobs require an explicit pool, apart from the public Development option for an eligible Trial described below. Legacy execution can retain an earlier worker only when the server verifies the exact published chain, configuration and permitted membership. Test and Stage consume the plan's Development bucket; Production uses the Production bucket. Pool membership does not replace capacity, online status, profile compatibility, quotas or concurrency.

Public Trial: platform-managed Development pool

An eligible public Trial can automatically use the platform-managed public Development pool when no explicit Development policy exists. An explicit policy, including an empty one, takes precedence. This grants neither private workers nor permission to save pools; availability and plan limits still apply. See [workers and capacity](#).

Subscription recovery

The funding projection uses Billing's canonical period representation and is reconciled with the authorized subscription. Correcting Subscription funding projection does not match does not change the contractual period, cycle, granted credits or consumption, and does not extend a Trial. Tenant, contract and funding checks remain enforced. For a persistent error, review the existing operation and subscription before repeating registration or checkout.

Developer Tools: CLI and personal keys

Verify distributions and automate only authorized scope.

Developer Tools provides platform-specific CLI distributions, checksums, signature information, instructions and release notes. It also exposes authorized personal-key management. The CLI version can differ from the server version. A version label does not replace integrity verification or grant additional permissions.

Before you begin

Use an authorized laboratory computer and a personal key with a defined purpose and lifetime. Keep keys out of shared arguments, screenshots, repositories and reports. Authorization remains tied to identity, tenant and permissions; the CLI does not bypass UI rules.

Procedure

1. Open Developer Tools or Downloads in the account menu and choose your operating-system distribution.
1. Read version, checksum, signature status and instructions. Download to an authorized local location and compare SHA-256 exactly with the published value.
1. Install as instructed and run `dova --version`. Check which executable PATH resolves if older installations exist.
1. Create a personal laboratory key where permitted and retain it through protected custody. Never include its value in documentation.
1. Run `dova login --url` with your authorized laboratory service and enter the key through the interactive prompt. Check returned identity and tenant.
1. Start with status and filtered Repository, Mapping or run lists. Compare identifiers with the UI before automating writes.
1. When finished, distinguish `dova logout` from server-side revocation. Logout clears local authentication; revocation prevents further use of that key on other devices.

Verify the result

The checksum matches and the intended CLI reports its version. A read-only query shows the expected identity and objects. Inspect command help for the installed distribution rather than assuming every option matches another version.

Errors and recovery

A checksum mismatch blocks using the artifact until its origin is verified. For 401/403, review key validity, expiry and permissions. A timeout can leave server work running: inspect the existing attempt before retrying. Protect the local profile because it can retain authentication.

Synthetic example

On your authorized laboratory host, run `dova --version` and `dova status --output json`. Do not place a key directly in a command line or treat an example domain as a real service.

Verification scope

Use synthetic data and the role authorized for the task. Compare a visual confirmation with the saved record or relevant attempt. Retain only safe identifiers and messages for support; do not share secrets, personal information or tenant data. If an operation outcome is uncertain, check its state before repeating it.

GitOps kit per workspace

The [GitOps kit 1.1.2](#) binds Tenant, Domain and Environment for queries, draft reservations, publications and leases. Use `source.environment` and `source.domainId` for exports; `target.environment` and `bindings.domainId` for imports. Defaults are Development and General; preserve state per target. See the [Finance example and state recovery](#). Direct Production authoring requires governed promotion rather than the import command.

Account menu: context, appearance and sign-out

Confirm identity and scope before acting in the platform.

The account menu shows identity, workspace and active role. It also offers personal actions such as Profile, Downloads, API Keys, Documentation and Sign out according to available access. Light/Dark and palettes change presentation; they do not change permissions or make an account an administrator of other tenants.

Before you begin

Know the workspace where you should work and use only contexts offered to your identity. Review Open items and unsaved changes before changing tenant or signing out. Do not use a context switch to resolve uncertainty about draft ownership.

Procedure

1. Open the menu and read identity, tenant and active role. Stop write actions if the context is unexpected.
1. Inspect Open items and deliberately save or discard necessary changes. An open editor does not guarantee persistence.
1. Select only an authorized workspace offered by the UI. Wait for transition and verify context again.
1. If permitted role selection is offered, choose the intended role and check resulting actions. A hidden action can be correct behavior.
1. Adjust Light/Dark or an available palette. Check readability, focus and states; color should not be your only reference.
1. Open Profile for preferences and personal security or Developer Tools for downloads and automation. Use Documentation for product help.
1. When finished on a shared computer, use Sign out. Return through normal login and MFA. Closing a tab is not explicit logout.

Verify the result

You can identify the tenant of displayed records and distinguish appearance from authorization. Perform a reversible exercise by changing theme and restoring the original without changing data. Check that keyboard focus and text remain visible.

Errors and recovery

If switching fails, verify the actually active context before retrying. If an object disappears, check tenant, role, filters and permission rather than immediately recreating it. If a personal action is unavailable, inspect authorized access instead of relying on a role name.

Synthetic example

In Training, open Profile from the menu, inspect timezone and close without changes. Then enable Dark and return to Light; neither action should publish or execute a Workflow.

Verification scope

Use synthetic data and the role authorized for the task. Compare a visual confirmation with the saved record or relevant attempt. Retain only safe identifiers and messages for support; do not share secrets, personal information or tenant data. If an operation outcome is uncertain, check its state before repeating it.

Tenant, Domain and Environment in the user menu

The Tenant, Domain and Environment summary appears in the bottom operational status bar of Dova and Administrator. Click the summary to open the user menu with all three selectors. It retains the environment badge and color without a top strip or duplicate fields.

Open the user button in the left sidebar. In the same menu, choose Tenant, Domain and Environment from the authorized values. Wait for the context to load and check the environment before editing or running. These three selectors replace the large context section on the screen. The menu opens to the right of its button when space allows and fits the visible viewport; on small screens, scroll inside it to reach Sign out.

After a Domain is successfully created, renamed or deleted, the Domain list updates without reloading the entire application. If the selected Domain is deleted, Dova checks an available destination while preserving the unsaved-change guard. If switching cannot complete, review the context and use Retry context; do not continue writing in an unavailable context.

Interface language

In the user menu, select ES, EN or PT to change the interface language. The preference is saved to your profile and used by Dova and Administrator. If saving fails, the interface reports it and restores the previous choice. Available application labels change; artifact names, identities, fields, code and user data are not renamed. Style, palette and permissions retain their own configuration.

Compact bottom bar

The bar keeps Tenant, Domain and Environment visible alongside connection state. Operational details opens a panel above the bar with Active, Workers, Queue, Plan, Autosave and Updated. Read operational details there without another permanent row. Validation state keeps its access when an action is relevant; DovaAI has its own space when docked. On small screens some text is abbreviated; context and accessible control names are retained.

Profile: preferences, security and personal requests

Manage your account without confusing it with user administration.

Profile contains personal information, preferences, security controls and privacy requests for the authenticated identity. It differs from Administrator → Users and does not grant roles or access to other tenants. Open it from the account menu and verify identity before editing. Deletion requests follow the stated process; they do not immediately remove shared objects or retained records.

Before you begin

Use a laboratory account for reversible preference checks. Before changing a password, prepare the authentication factors needed to sign in again. Treat a profile export as personal information and store it in an authorized location.

Procedure

1. Open Profile and inspect name, email, tenant and role. If identity is unexpected, close it and resolve context before editing.
1. Choose a supported Timezone and use Save preferences. Reopen to verify persistence. This preference does not change a Workflow calendar timezone.
1. For Change photo, use PNG, JPEG or WebP up to 10 MB. The flow center-crops to a 256-pixel square and optimizes the image before upload. Wait for confirmation.
1. Use Remove photo only when removal is intended and confirm the accepted result. A processing failure does not prove a photo was saved or removed.
1. To change a password, supply Current password, New password and Confirm new password. The UI requires 12–128 characters and the server can apply additional rules.
1. Remember that changing password signs out browser sessions and trusted devices. Authenticate again through the normal flow.
1. Distinguish Download profile export, Request profile data deletion and Request account deletion. Read confirmation before Submit request.

Verify the result

Change a laboratory account timezone, save and reopen to verify it, then restore the baseline. Privacy dialogs can be inspected and cancelled without submitting a request. Track a submitted request as pending until the process confirms its outcome.

Errors and recovery

If Save fails, check persisted state before retrying. If a session ends after password change, sign in again rather than assuming object loss. Correct a rejected photo format or size. Keep exports, MFA factors and recovery material out of shared screenshots.

Synthetic example

Temporarily change Timezone to America/Lima in a laboratory account. Check a supported time presentation and restore the original preference without changing Workflow scheduling.

Open items: editors, drafts and navigation

Keep open, saved and published state distinct.

Open items collects the Mapping, Session and Workflow editors open in the workspace. Each tab represents an editing context; it is not a published version or a running job. Name, type and pending-change marker support navigation, but object identity, tenant and version remain the reference.

Before you begin

Open related synthetic objects with distinct names. Confirm tenant and location before working across several tabs. Understand the difference between saving a draft, publishing a version, executing and discarding a local change. Navigation and memory optimizations do not replace Save.

Procedure

1. Open the Mapping, Session and Workflow for a laboratory process. Identify each tab by name and type.
1. Switch tabs and return to the editor. Continuing an edit does not prove it was saved or published.
1. Make a reversible edit and inspect the pending-change marker. Save if intended or revert the change; wait for the accepted result.
1. Open a tab context menu and read whether an action affects one item, several items or other tabs. Avoid bulk closing when needed changes remain.
1. When closing a changed editor, deliberately choose Save, Discard or Cancel from the presented options. Cancel preserves context; discarding local changes does not delete the saved object.
1. Reopen from Repository or its manager and check identifier, saved content and version. Only publication establishes a published version.
1. Before changing tenant, signing out or finishing, review pending items and resolve each deliberately.

Verify the result

Modify a synthetic description, switch to another tab, return and cancel closing. Verify the draft remains present. Then revert or save the change, close and reopen to confirm expected persisted state. This exercise does not require executing the Workflow.

Errors and recovery

If an editor differs from expectations, check type, identifier, tenant and version before changing it. A closed tab does not prove deletion, and a visible draft does not prove publication. On compact screens use keyboard focus and offered menus. Investigate loading errors before repeating actions.

Synthetic example

Open m_CustomerNames and s_CustomerNames in Training. Change a laboratory description, visit the Session and return to the Mapping; explicitly save or discard before closing.

Environment context

Before using this module, confirm Tenant, Domain and Environment in the user menu. The initial environment is Development. Module, Domain and environment capabilities apply together; Production is read only without explicit authorization. Switching the selector does not move artifacts or publish or execute their data. See [environments and recovery](#) and [environment promotion](#).

DovaAI: read-only diagnosis and guidance

Get a confirmed cause, the right component and brief steps that you carry out.

DovaAI uses the authorized tenant, module and current view. It follows the language of your latest message and gives brief recommendations about components, settings and executions. Conversation history continues across module changes within the permitted context.

Before you start

Confirm tenant and permissions. Configure the provider through its protected procedure. Never paste keys into chat. Queries consume effective plan and user allowances; starting another conversation does not reset usage.

Ask for a concrete solution

Describe the field, input and expected result. For example: "Replace exactly jorge with Jorge in name". Use Replace values with field=name, find=jorge, replacement=Jorge, matchMode=exact and caseSensitive=true. This rule does not capitalize other names. To keep only selected columns, use Field selector and its fields list.

If you attach a supported synthetic sample, context uses schema metadata. The attachment does not establish a permanent execution input or authorize edits. Inspect column names before sharing them: headers can also contain sensitive information.

Diagnose Workflow and Session failures

Open the failed execution and ask for its cause. DovaAI reads safe evidence from the authorized attempt: executed versions, profile, phase, limits and available counters. It separates confirmed causes from hypotheses. A generic error without original details does not alone prove a destination collision or insufficient memory.

Expect a short answer: confirmed problem, exact screen path and values, followed by a few manual steps. For example, when verified configuration leaves inherited timestamps disabled: Sessions → Session → Runtime → Target filename timestamp → Enabled; retain the template, save and publish. Update Workflow versions afterwards when needed.

Who makes changes

DovaAI only reads and recommends. It never creates, edits, configures, applies, publishes, runs or deletes Mappings, Sessions, Workflows, Connections or promotions, even when requested or confirmed. Guidance points to the next control; you decide and use the normal platform controls.

Verify the result

Compare the diagnosis with the log and actual executed version. After your edits, deliberately save and publish, update dependencies when needed and run an authorized test with an identifiable output. An assistant response does not prove that configuration changed or execution completed.

A concise recommendation format

DovaAI presents the recommended component, a one-line flow and steps with properties and values as code. It explains the effect on other fields without unnecessary navigation. For example:

Connect two Clean text components in series:

```
Raw file → Clean text 1 → Clean text 2 → Target
```

1. Clean text 1: Fields = `first_name` ; Operations = `trim` .
2. Clean text 2: Fields = `last_name` ; Operations = `upper` .

Both fields travel together. Each component changes only its selected field and preserves the others. Wording may vary: check names and values against the current component. The assistant recommends; you configure and execute.

Environment context

Before using this module, confirm Tenant, Domain and Environment in the user menu. The initial environment is Development. Module, Domain and environment capabilities apply together; Production is read only without explicit authorization. Switching the selector does not move artifacts or publish or execute their data. See [environments and recovery](#) and [environment promotion](#).

Conversation and environment

DovaAI uses the authorized Tenant, Domain and Environment and selected artifact. Switching context cancels earlier requests and starts a conversation for the new environment. A pending reply from the earlier context is not added to the new conversation. The server verifies environment identity and versions; form values, records and credentials are excluded from context.

Move and dock DovaAI

Drag the icon with a mouse or finger: the launcher follows the pointer within the visible area. As it approaches the bottom bar, a preview shows the reserved space. Release it there to dock with a snap animation. Drag it out of the bar to detach with a brief animation and place it elsewhere on the screen.

You can also move the window by its header. Dragging the header to the bottom bar minimizes the chat and docks its launcher; reopening it preserves the conversation in the same context. The reduced-motion preference skips these animations.

With the icon focused, Alt + End docks it, Alt + arrow keys move and undock it, and Alt + Home restores its position. The position is saved in this browser for the authenticated user and tenant; it is not shared with other identities.

Select context and a screenshot

1. Open the chat. Select element and Select area sit together at the bottom left of the message box. Use the first to point to a component and the second to draw a rectangle over Dova or Administrator.
2. Check the outline. Arrow keys move the pointer or area; in area mode, Shift + arrow keys change its size. Enter or Capture selection confirms; Escape or Cancel cancels. Selection does not activate the controls it points to.

3. Review the preview before Send. It contains a real screenshot of the area and a summary of its visible elements. Dova excludes form values and elements marked private from the prepared screenshot. Remove the image if it is not the area you want to share.

Images in the message

Paste an image or drop a PNG, JPEG or WebP into the chat composer. An attachment button is not needed. Review and remove each preview before Send. Up to four images are allowed per message; they are prepared within 2 MiB and 2048 pixels per side. Images you paste may contain sensitive information: review their contents before sending. DovaAI remains read only and the query uses the effective plan allowance. An identity or context change discards the previous submission.

Conversation language and guidance

Conversation controls follow the message language in Spanish, English or Portuguese. A greeting or a thank-you does not add an unrelated Show me button. Guidance appears when relevant to a Dova task or when requested; its steps remain limited to the authorized screen and permissions.

Open the chat and write a message

When the chat opens, Preparing Dova AI... appears and its controls remain disabled until loading finishes and the authorized context is verified. Wait for the controls to become available before writing or selecting a screenshot. If preparation fails, check the notice, context and availability, then close and reopen the chat.

The text field uses the full width. Below it, inside the same box, the two context selectors sit together on the left and the Send button, with an upward arrow, sits on the right. Activity is shown while sending; a failure keeps the message so you can review and retry it.

Administrator overview

Read administrative status before changing the platform.

Before you begin

Confirm account, tenant and active role in the account menu. Inspection requires view permission; changing records requires the specific management permission. A hidden or disabled action can reflect authorized scope; do not switch tenants to evade a denial. Work only with records in the workspace you are allowed to administer.

Procedure

1. Open Overview and inspect Tenants, Users, Subscriptions, Invoices, Usage and Security.
2. Select a worker and a 1h, 6h or 24h interval to compare CPU, memory and disk.
3. Open the relevant module to investigate its underlying record.

Behavior and limits

Administrator has ten primary modules; the sixteen chapters in this manual also cover specialized procedures. Charts represent available observations and actions depend on the active role.

A card does not prove payment, execution success or guaranteed capacity. An absent measurement or a failed load is not zero.

Compare samples for the same worker and period. Inspect measurement coverage before attributing a change to a pipeline. Zoom controls change the visible range; Refresh requests current observations.

Verify the result

After saving or confirming, reopen the detail and check the state reported by the server. Distinguish an accepted request from a completed operation and retain only minimal evidence without sensitive information. When an operation is pending, wait for its final state in the corresponding view before starting another action on the same object.

Laboratory example

Use the synthetic tenant `documentation-lab` and, where relevant, worker `worker-demo-a`. Note the previous state, perform one authorized action and compare the resulting state. These names illustrate the procedure; they do not identify production records or authorize creating cloud resources. For a review without changes, inspect the detail and cancel any open form.

If something fails

If a conflict appears, refresh the inventory and review current values before repeating. If the session expired, sign in and complete MFA when required. For a permission error, check the role and membership; for a provider error, inspect the operation and its safe message. Do not alter storage, revisions or controls to force an outcome.

Related reading

- [Administrator](#)
- [Preferences](#)
- [Audit](#)

Continuous updates and stable reading

Overview checks for changes every five seconds while the view is active. Charts and cards retain their structure and previous data while a response arrives; selection, range, zoom and scroll are preserved. Live shows the last confirmed update. Last confirmed and Refresh pending indicate that the read could not be renewed: visible values remain the last observation, not a new measurement.

The screen refresh interval does not change telemetry sampling frequency. A missing sample is shown as unknown rather than converted to zero. Selection controls and legends retain their position while values update.

Dova base server

Platform administrators see `vm-clouddata-pe` labeled Base server. It is the server hosting Dova and remains in Overview even when samples are missing or its state cannot be confirmed. It is excluded from execution worker counts and has no removal action in this view.

Metrics come from local host sampling when its identity can be verified; they do not replace a GCP state query. If identity or measurements are unavailable, the row remains and displays unknown information rather than attributing another host's activity to it.

Fast Administrator reads

Administrator loads authorized module or tab data when needed and briefly reuses reads in RAM with size and expiry limits. Identical overlapping reads are joined; changes to data, identity, role or context invalidate the corresponding reuse. An earlier context's response does not replace the current view. Periodic updates preserve selection and visible state while data remains valid. These measures reduce repeated reads; actual timing depends on data, network and provider.

Grid row updates preserve open menus, filters, selection and scroll. When an action starts, its control shows In progress and other actions in that grid are blocked until its result; other modules and reading controls remain available.

Navigation shared with Dova

Dova and Administrator share the left sidebar's presentation and behavior: text, active and hover states, keyboard focus, collapse and user alignment. The user menu opens beside its button and remains within the screen. Available modules and actions still depend on your permissions.

Read CPU, memory and disk

CPU, Memory and Disk show Not available when there is no valid current measurement, the VM is unavailable or heartbeat/provider observation has expired. An unavailable reading is not zero. Earlier history may remain with Historical or Last recorded and its timestamp; do not treat it as current usage.

Available disk represents free space. Review Capacity, Used and the percentage to understand occupied space. The vertical axis starts at zero and uses known capacity for scale; the horizontal axis uses the displayed local time. Hover, focus and keyboard reveal the exact sample and, when capacity is known, used and total values. Last sample and timezone identify when the data was observed.

The Dova base server row identifies the platform host and remains protected. Its local telemetry is used only after identity verification; those readings do not by themselves confirm native cloud VM state.

Cloud workers

Provision external capacity, inspect actual VM state and manage its actions from the grid.

Before you begin

Confirm account, tenant and active role in the account menu. Inspection requires view permission; changing records requires the specific management permission. A hidden or disabled action can reflect authorized scope; do not switch tenants to evade a denial. Work only with records in the workspace you are allowed to administer.

Procedure

1. In Cloud workers use Connections to configure and verify the cloud account and authorized regions.
2. In Profiles define vCPU, Memory and Disk.
3. In Workers choose Provision worker, review tenant, connection, region and profile, then confirm once.
4. Check Provisioning history for the outcome. In Workers use Refresh to update the provider observation.
5. Open More actions in the row or the DOVA_DATA_GRID context menu for authorized actions. The row shortcut opens View details; Columns lets you display the optional Runtime column.

Behavior and limits

There are four tabs: Workers, Provisioning history, Connections and Profiles. Google Cloud, AWS and Azure are connection types; each operation depends on actual provider capabilities and your permissions.

Status shows the VM state observed in the cloud: Running, Stopped or Paused. In GCP, Paused corresponds to SUSPENDED. It can also show transitions such as Starting or Stopping. Unknown or Unavailable means there is no verifiable observation; Last known identifies an earlier state that could not be refreshed. These states do not establish a recent heartbeat or available capacity.

The optional Runtime column shows the worker process and job admission: Online, Offline, Draining or Disabled. Enable it from Columns. It is independent of VM state: Running does not guarantee that the worker is Online. Pause scheduling stops new assignments and lets active work finish; Resume scheduling enables assignments again. These actions do not suspend or resume the provider VM.

The row menu and controls beside Worker name offer actions according to observed state and authorization. Start VM starts a Stopped VM and Resume VM resumes a Paused VM when supported by the provider. A default, busy or public worker can block Stop VM. A public worker shares capacity; it never makes its consumer Session or data public.

Reset VM forcibly restarts a Running GCP VM and requires confirmation. Before submitting it to the provider, the server checks again for active jobs, runs and reservations across every tenant authorized to use that worker. A default or public worker can be reset when this complete check confirms it is idle; its references and sharing policy are retained. The action appears only for compatible providers and states.

While a Reset has no confirmed outcome, Dova blocks new assignments and another Reset. The main status still describes the VM; Reset pending or Check reset reports the operation separately. Use Refresh to check its result without submitting it again. Only a completed native operation and a valid observation can resolve the block; a Running VM alone does not prove Reset completion. A confirmed Reset does not guarantee process readiness: Runtime waits for the next heartbeat.

Inspect Tenant, Public, Status, Slots and Last seen. Configured and available slots can differ because of physical admission. A price estimate is neither an invoice guarantee nor confirmation of available capacity.

GCP worker networking

New GCP workers are provisioned with private IP addresses only; no public IP selector is offered. Configure Cloud NAT in the subnet region so the worker can reach Dova over HTTPS and download packages and dependencies from Debian, PyPI and GitHub. Private Google Access supports access to Google APIs, but does not replace Cloud NAT for those external destinations. Also check DNS and the outbound rules required by authorized connectors.

For maintenance, use IAP SSH with OS Login when the required IAM roles and firewall rules are configured. The worker pulls jobs and sends heartbeats through outbound connections; it does not need an inbound application port to receive them.

The `public_shared` policy shares capacity among authorized tenants; it is independent of whether the VM has a public IP address. Preserve isolation for each job and the scope of its credentials and data.

Verify the result

After saving or confirming, reopen the detail and check the state reported by the server. Distinguish an accepted request from a completed operation and retain only minimal evidence without sensitive information. When an operation is pending, wait for its final state in the corresponding view before starting another action on the same object.

Laboratory example

Use the synthetic tenant `documentation-lab` and, where relevant, worker `worker-demo-a`. Note the previous state, perform one authorized action and compare the resulting state. These names illustrate the procedure; they do not identify production records or authorize creating cloud resources. For a review without changes, inspect the detail and cancel any open form.

If something fails

If a conflict appears, use Refresh and review current values before starting another action. If Reset pending or Check reset appears, use Refresh to inspect the operation; do not repeat Reset to resolve it. If the provider confirms a failure, inspect its safe message and the refreshed state before deciding on another action. If the session expired, sign in and complete MFA when required. For a permission error, check the role and membership. Do not alter storage, revisions or controls to force an outcome.

Related reading

- [Administrator](#)
- [Preferences](#)
- [Audit](#)

Provider-synchronized VM controls

In the row menu, Pause VM suspends a Running GCP VM and produces Paused (SUSPENDED). Resume VM resumes a Paused VM; Start VM starts a Stopped VM and Stop VM stops it. Reset VM restarts a Running VM while retaining its registration. Dova verifies current state and ownership before sending an action and checks the native result; an accepted request is not a completed operation.

Pause VM differs from Pause scheduling: the former suspends the VM; the latter stops job admission and lets active work finish. After Start, Resume or Reset, Runtime needs an authenticated heartbeat to confirm the process is Online. Incompatible actions are disabled during an operation or while jobs and dependencies prevent it.

GCP provides these native controls according to state. Azure offers Start, Stop and Deprovision; Pause and Reset are not presented as equivalent operations. AWS can appear as a connection type, but its VM lifecycle is unavailable in this edition.

Remove a VM and recover assignments

Open Remove VM and unlink, review resource identity and impact on Sessions, executions and pools, then confirm the intended deletion. For a supported owned VM, manually dismantling every reference or first powering it off is not a prerequisite for this operation. Removing capacity can interrupt active executions; schedule the operation to suit your needs.

After confirming provider absence, Dova retires the active worker, clears Runtime references and shows Unassigned for Session bindings that used it. Definitions, published versions, configuration and audit history are preserved. If the VM is deleted directly in the provider, reconciliation applies the same treatment after verifying absence. A provider-read error is not evidence that the VM was deleted.

Before running again, select compatible authorized capacity and review the environment pool. Unassigned does not silently choose another worker. You can provision a new VM and assign it explicitly. Permissions, resource ownership, the protected base server and pending operations remain checked; removal does not bypass these protections.

Environment context

Before using this module, confirm Tenant, Domain and Environment in the user menu. The initial environment is Development. Module, Domain and environment capabilities apply together; Production is read only without explicit authorization. Switching the selector does not move artifacts or publish or execute their data. See [environments and recovery](#) and [environment promotion](#).

Environment worker pool

In Configuration → Dova Integration Runtime, Environment worker pool identifies the current environment. Allowed workers assigns existing workers; Default worker must belong to that selection. Pool policy starts as Dedicated. Select Shared only when you explicitly want to share capacity and every participating pool also declares Shared. A matching name does not share a physical worker.

Save environment pool changes only the selected environment's policy and neither starts a job nor changes the tenant's global default. The review uses a configuration fingerprint; if another user changes it, Refresh pool lets you reload and review before saving. Controls require configuration and environment Runtime administration. Built-in is not offered as an execution worker.

Without an explicit pool, Production does not borrow Development capacity and new jobs do not run through an implicit default. Legacy execution can retain an earlier worker only when the server verifies the exact published chain, configuration and permitted membership. Test and Stage consume the plan's Development bucket; Production uses the Production bucket. Pool membership does not replace capacity, online status, profile compatibility, quotas or concurrency.

Before stopping or retiring capacity

Pause VM and Stop VM retain active-work, default-worker and shared-capacity checks; review references and impact before using them. Pause scheduling stops new assignments and lets current work finish. To retire the VM permanently, use Remove VM and unlink and review its confirmation: after verifying cloud absence, Dova clears assignments and shows Unassigned without deleting definitions or published versions. Assign compatible capacity explicitly before running again.

Actions require the module permission and the relevant environment's Runtime capability, or an explicitly granted global permission. A shared VM requires authorization across all its environments.

Resources and permissions per environment

Tenant and Environment selected in the user menu scope Cloud workers, their cloud connections, provisioning profiles, operations and dependency backups. A Domain declared on a resource must also match. Overview, resource history and operational Audit events respect this scope. Users, roles, tenants and billing retain their own authority: they are not independent copies per environment.

Managing a VM requires the module permission plus `etl.environments`.

`<environment>.runtime.manage`, or the global Runtime permission when explicitly granted. If multiple environments share the VM, you need authorization in every affected environment. Switching the selector neither moves a VM nor expands permissions.

Compact controls follow cloud state

Beside Worker name, the Pause, Start/Resume, Stop and Reset icons show VM actions. Read each icon's tooltip. Start is available for a Stopped VM and Resume for a Paused VM when the provider and your permissions allow it. The worker process being Offline does not by itself prevent starting a powered-off VM.

Dova checks current cloud state and authorized actions. A stale observation, pending operation, active work or permission restriction can disable a control. After Start, Resume or Reset, wait for the provider result and the worker heartbeat; VM Running and Runtime Online are separate checks. If the result is uncertain, use Refresh before repeating the action.

Observe provider changes

GCP provides Start, Stop, Pause/Suspend, Resume, Reset and VM removal according to state. Azure provides Start, Stop/Deallocate and removal; Pause/Resume/Reset are not presented as equivalent operations. AWS VM lifecycle is not enabled in this edition. Visible capabilities depend on the provider and effective authorization.

Provider synchronization is periodic. Changes made directly in the cloud appear after a new valid observation; this is not an instant notification across the fleet. Check observation time and use Refresh to query state. An accepted operation does not prove completion, and a Running VM still needs a heartbeat for Runtime Online.

Users and access

Manage identities, memberships and roles within the authorized scope.

Before you begin

Confirm account, tenant and active role in the account menu. Inspection requires view permission; changing records requires the specific management permission. A hidden or disabled action can reflect authorized scope; do not switch tenants to evade a denial. Work only with records in the workspace you are allowed to administer.

Procedure

1. Open Users, find the account and choose View details.
2. In New user or Edit user enter Name, Email, Tenant and the appropriate status.
3. Use Select roles to assign the required responsibilities; save and inspect the record again.

Behavior and limits

Identity status, membership status and access readiness are different concepts. Changing memberships requires platform administration. An actor cannot delegate permissions it does not possess.

These actions cannot disable or demote your own account, or remove the last active platform administrator. Do not publish activation links or recovery material.

Also inspect Access and Last login. Never means no observed sign-in, not a failed execution. Presence does not grant access, and an active account may still require activation or MFA.

Verify the result

After saving or confirming, reopen the detail and check the state reported by the server. Distinguish an accepted request from a completed operation and retain only minimal evidence without sensitive information. When an operation is pending, wait for its final state in the corresponding view before starting another action on the same object.

Laboratory example

Use the synthetic tenant `documentation-lab` and, where relevant, worker `worker-demo-a`. Note the previous state, perform one authorized action and compare the resulting state. These names illustrate the procedure; they do not identify production records or authorize creating cloud resources. For a review without changes, inspect the detail and cancel any open form.

If something fails

If a conflict appears, refresh the inventory and review current values before repeating. If the session expired, sign in and complete MFA when required. For a permission error, check the role and membership; for a provider error, inspect the operation and its safe message. Do not alter storage, revisions or controls to force an outcome.

Related reading

- [Administrator](#)
- [Preferences](#)
- [Audit](#)

Identity and environments

Tenant and Environment selected in the user menu scope Cloud workers, their cloud connections, provisioning profiles, operations and dependency backups. A Domain declared on a resource must also match. Overview, resource history and operational Audit events respect this scope. Users, roles, tenants and billing retain their own authority: they are not independent copies per environment.

Tenants and capacity

Separate operational workspaces, access and commercial terms.

Before you begin

Confirm account, tenant and active role in the account menu. Inspection requires view permission; changing records requires the specific management permission. A hidden or disabled action can reflect authorized scope; do not switch tenants to evade a denial. Work only with records in the workspace you are allowed to administer.

Procedure

1. Open Tenants and inspect the workspace through View details.
2. For an authorized change, use New tenant or Edit tenant and complete Name, Type, Parent reseller, Status and permitted limits.
3. Then check the effective subscription.

Behavior and limits

The form distinguishes Organization, Individual and Reseller. Switching tenant revalidates active membership. A plan name alone does not describe all contracted entitlements. Creating a tenant and creating a user are separate operations.

Commercial expiry preserves objects and permits the supported safe reads. Commercial reactivation does not override a security suspension. Current values come from the applicable plan and frozen contract.

Separate Seats, Monthly jobs and Storage MB: these resources have different units. For an expired trial or restricted commercial state, inspect the notice and Plans & billing without assuming object creation, editing or execution is allowed.

Verify the result

After saving or confirming, reopen the detail and check the state reported by the server. Distinguish an accepted request from a completed operation and retain only minimal evidence without sensitive information. When an operation is pending, wait for its final state in the corresponding view before starting another action on the same object.

Laboratory example

Use the synthetic tenant `documentation-lab` and, where relevant, worker `worker-demo-a`. Note the previous state, perform one authorized action and compare the resulting state. These names illustrate the procedure; they do not identify production records or authorize creating cloud resources. For a review without changes, inspect the detail and cancel any open form.

If something fails

If a conflict appears, refresh the inventory and review current values before repeating. If the session expired, sign in and complete MFA when required. For a permission error, check the role and membership; for a provider error, inspect the operation and its safe message. Do not alter storage, revisions or controls to force

an outcome.

Related reading

- [Administrator](#)
- [Preferences](#)
- [Audit](#)

Environment context

Before using this module, confirm Tenant, Domain and Environment in the user menu. The initial environment is Development. Module, Domain and environment capabilities apply together; Production is read only without explicit authorization. Switching the selector does not move artifacts or publish or execute their data. See [environments and recovery](#) and [environment promotion](#).

Roles and permissions

Group capabilities by responsibility and tenant.

Before you begin

Confirm account, tenant and active role in the account menu. Inspection requires view permission; changing records requires the specific management permission. A hidden or disabled action can reflect authorized scope; do not switch tenants to evade a denial. Work only with records in the workspace you are allowed to administer.

Procedure

1. In Roles & permissions open Roles and View details.
2. Inspect Permissions to identify the required capability.
3. In New role or Edit role define Key, Label and minimum permissions; save and verify access with an authorized laboratory account.

Behavior and limits

The catalog describes permissions but does not prove they are assigned to the user. Platform roles belong to platform scope and custom roles remain bound to their tenant.

The unique master administrator has a specific verification for the DovaAI quota exemption. Not every administrator receives unlimited access. A visible permission cannot bypass tenant boundaries or authorize additional unrestricted superusers.

Check the effective assignment in Users after saving. The master DovaAI exemption requires the active persisted identity, active platform tenant, valid membership and selected role; provider readiness is still required.

Verify the result

After saving or confirming, reopen the detail and check the state reported by the server. Distinguish an accepted request from a completed operation and retain only minimal evidence without sensitive information. When an operation is pending, wait for its final state in the corresponding view before starting another action on the same object.

Laboratory example

Use the synthetic tenant `documentation-lab` and, where relevant, worker `worker-demo-a`. Note the previous state, perform one authorized action and compare the resulting state. These names illustrate the procedure; they do not identify production records or authorize creating cloud resources. For a review without changes, inspect the detail and cancel any open form.

If something fails

If a conflict appears, refresh the inventory and review current values before repeating. If the session expired, sign in and complete MFA when required. For a permission error, check the role and membership; for a provider error, inspect the operation and its safe message. Do not alter storage, revisions or controls to force

an outcome.

Related reading

- [Administrator](#)
- [Preferences](#)
- [Audit](#)

Environment context

Before using this module, confirm Tenant, Domain and Environment in the user menu. The initial environment is Development. Module, Domain and environment capabilities apply together; Production is read only without explicit authorization. Switching the selector does not move artifacts or publish or execute their data. See [environments and recovery](#) and [environment promotion](#).

Environment permissions

Authorization combines module permissions, Domain access and environment capabilities. The current catalog contains 23 environment permissions: `etl.environments.view`, `etl.environments.{development,test,stage,production}.{view,edit,publish,run,runtime.manage}`, `etl.environments.promote` and `etl.environments.runtime.manage`. Global view and Runtime grants retain their explicit scope; a grant for one environment does not grant access to all others. Module management does not replace these capabilities.

ETL Developer can work in Development, Test and Stage and has no Production access by default. Release Manager can review all four environments and publish into Test, Stage and Production for authorized promotions; this does not grant Production editing or execution. Promotion retains preview, exact published dependencies and explicit Production approval. Custom roles can grant the required capabilities per environment. A user identity is not duplicated per environment; check the active role, membership and effective permissions.

Production Mapping, Session and Workflow editors remain read only: accepted versions arrive through promotion. Configuring existing or new Production Connections is a separate capability requiring `etl.connections.manage` and `etl.environments.production.edit`; it does not unlock those three editors or legacy read-only Connections.

Subscriptions, DPU and billing

Understand consumption and verify current commercial terms.

Before you begin

Confirm account, tenant and active role in the account menu. Inspection requires view permission; changing records requires the specific management permission. A hidden or disabled action can reflect authorized scope; do not switch tenants to evade a denial. Work only with records in the workspace you are allowed to administer.

Procedure

1. Open Subscription & billing and select Tenant.
2. Review Subscription and the billing profile.
3. Use DPU & calculator for estimates and Payments & invoicing for permitted merchant configuration.
4. Inspect Licenses, Commerce, Global catalog and Tenant offers according to your responsibility.

Behavior and limits

The seven tabs separate contracts, usage, payments and catalog. DPU measures capacity; DPU-h multiplies capacity by active time. Development and Production have separate funding and concurrency. Tasks aggregate usage; the Workflow is not charged again.

Queued work and structural validation are not billable. Purchased concurrency does not create workers. Accepted DovaAI queries have tenant and user quotas separate from DPU. A browser return or manual invoice status does not prove confirmed payment.

Calculation example: 4 DPU for 15 minutes consumes 1 DPU-h. Credits follow verified funded periods, not necessarily calendar months. A payment or fiscal-document sandbox does not establish production acceptance.

Verify the result

After saving or confirming, reopen the detail and check the state reported by the server. Distinguish an accepted request from a completed operation and retain only minimal evidence without sensitive information. When an operation is pending, wait for its final state in the corresponding view before starting another action on the same object.

Laboratory example

Use the synthetic tenant `documentation-lab` and, where relevant, worker `worker-demo-a`. Note the previous state, perform one authorized action and compare the resulting state. These names illustrate the procedure; they do not identify production records or authorize creating cloud resources. For a review without changes, inspect the detail and cancel any open form.

If something fails

If a conflict appears, refresh the inventory and review current values before repeating. If the session expired, sign in and complete MFA when required. For a permission error, check the role and membership; for a provider error, inspect the operation and its safe message. Do not alter storage, revisions or controls to force an outcome.

Related reading

- [Administrator](#)
- [Preferences](#)
- [Audit](#)

Environment context

Before using this module, confirm Tenant, Domain and Environment in the user menu. The initial environment is Development. Module, Domain and environment capabilities apply together; Production is read only without explicit authorization. Switching the selector does not move artifacts or publish or execute their data. See [environments and recovery](#) and [environment promotion](#).

Environment worker pool

In Configuration → Dova Integration Runtime, Environment worker pool identifies the current environment. Allowed workers assigns existing workers; Default worker must belong to that selection. Pool policy starts as Dedicated. Select Shared only when you explicitly want to share capacity and every participating pool also declares Shared. A matching name does not share a physical worker.

Save environment pool changes only the selected environment's policy and neither starts a job nor changes the tenant's global default. The review uses a configuration fingerprint; if another user changes it, Refresh pool lets you reload and review before saving. Controls require configuration and environment Runtime administration. Built-in is not offered as an execution worker.

Without an explicit pool, Production does not borrow Development capacity and new jobs do not run through an implicit default. Legacy execution can retain an earlier worker only when the server verifies the exact published chain, configuration and permitted membership. Test and Stage consume the plan's Development bucket; Production uses the Production bucket. Pool membership does not replace capacity, online status, profile compatibility, quotas or concurrency.

Funding projection and period

The funding projection uses Billing's canonical period representation and is reconciled with the authorized subscription. Correcting Subscription funding projection does not match does not change the contractual period, cycle, granted credits or consumption, and does not extend a Trial. Tenant, contract and funding checks remain enforced. For a persistent error, review the existing operation and subscription before repeating registration or checkout.

Administrative audit

Investigate actions within the permitted scope.

Before you begin

Confirm account, tenant and active role in the account menu. Inspection requires view permission; changing records requires the specific management permission. A hidden or disabled action can reflect authorized scope; do not switch tenants to evade a denial. Work only with records in the workspace you are allowed to administer.

Procedure

1. Open Audit and use search and filters to find the event.
2. In View details inspect action, actor, timestamp and related subject.
3. Open the corresponding screen to confirm its current state and record only the necessary evidence.

Behavior and limits

Events help correlate administrative changes. Pagination and columns change presentation, not the record. Exports must be limited to the authorized purpose.

An audit entry alone does not prove that a provider completed a VM operation, a payment settled or an execution produced the right result. Protect identities and sensitive content.

To investigate a role change, correlate the event with Users and Roles & permissions. For infrastructure inspect Cloud workers; for executions inspect Monitor. Preserve the distinction between the event's historical state and the object's current state.

Verify the result

After saving or confirming, reopen the detail and check the state reported by the server. Distinguish an accepted request from a completed operation and retain only minimal evidence without sensitive information. When an operation is pending, wait for its final state in the corresponding view before starting another action on the same object.

Laboratory example

Use the synthetic tenant `documentation-lab` and, where relevant, worker `worker-demo-a`. Note the previous state, perform one authorized action and compare the resulting state. These names illustrate the procedure; they do not identify production records or authorize creating cloud resources. For a review without changes, inspect the detail and cancel any open form.

If something fails

If a conflict appears, refresh the inventory and review current values before repeating. If the session expired, sign in and complete MFA when required. For a permission error, check the role and membership; for a provider error, inspect the operation and its safe message. Do not alter storage, revisions or controls to force an outcome.

Related reading

- [Administrator](#)
- [Preferences](#)
- [Audit](#)

Environment context

Before using this module, confirm Tenant, Domain and Environment in the user menu. The initial environment is Development. Module, Domain and environment capabilities apply together; Production is read only without explicit authorization. Switching the selector does not move artifacts or publish or execute their data. See [environments and recovery](#) and [environment promotion](#).

Registration review

Review pending requests without confusing approval with provisioning.

Before you begin

Confirm account, tenant and active role in the account menu. Inspection requires view permission; changing records requires the specific management permission. A hidden or disabled action can reflect authorized scope; do not switch tenants to evade a denial. Work only with records in the workspace you are allowed to administer.

Procedure

1. Open Registrations, find the request and inspect View details.
2. For an eligible pending request, use Approve registration or the rejection action, verify the applicant and complete the required reason.
3. Save and refresh the inventory before another decision.

Behavior and limits

Viewing requests and deciding them are separate permissions. The result records administrative review; tenant and user creation and readiness must be checked separately.

If another administrator already decided or the record changed, refresh and review its latest state. Do not announce an account is ready merely because the request is approved.

Inspect name, organization, type, received date and previous review. The reason should explain the decision without unnecessary data. A request that is no longer pending is not eligible for another ordinary decision.

Verify the result

After saving or confirming, reopen the detail and check the state reported by the server. Distinguish an accepted request from a completed operation and retain only minimal evidence without sensitive information. When an operation is pending, wait for its final state in the corresponding view before starting another action on the same object.

Laboratory example

Use the synthetic tenant `documentation-lab` and, where relevant, worker `worker-demo-a`. Note the previous state, perform one authorized action and compare the resulting state. These names illustrate the procedure; they do not identify production records or authorize creating cloud resources. For a review without changes, inspect the detail and cancel any open form.

If something fails

If a conflict appears, refresh the inventory and review current values before repeating. If the session expired, sign in and complete MFA when required. For a permission error, check the role and membership; for a provider error, inspect the operation and its safe message. Do not alter storage, revisions or controls to force an outcome.

Related reading

- [Administrator](#)
- [Preferences](#)
- [Audit](#)

Developer Tools, CLI and access

Download verified tools and limit credential scope.

Before you begin

Confirm account, tenant and active role in the account menu. Inspection requires view permission; changing records requires the specific management permission. A hidden or disabled action can reflect authorized scope; do not switch tenants to evade a denial. Work only with records in the workspace you are allowed to administer.

Procedure

1. Open Developer Tools and select the destination operating system.
2. Inspect published availability, version, checksum and signing information for the exact package.
3. Download, verify integrity and follow Quick start in an authorized environment.
4. In API Keys create only the necessary scope and revoke the key when no longer needed.

Behavior and limits

The CLI package version can differ from the Dova version. Downloading a tool does not grant permissions. Keys retain their authorized tenant and role scope; they do not replace server checks.

Do not invent platform availability, URLs or signatures. Do not put keys in documentation, terminal history or source. A revoked or unverifiably bound key must not be recovered through manual storage changes.

Distinguish availability, integrity and signing: they are different properties. The destination machine may use a different operating system from the browser. Check command scope before execution and use instructions bundled with that exact package.

Verify the result

After saving or confirming, reopen the detail and check the state reported by the server. Distinguish an accepted request from a completed operation and retain only minimal evidence without sensitive information. When an operation is pending, wait for its final state in the corresponding view before starting another action on the same object.

Laboratory example

Use the synthetic tenant `documentation-lab` and, where relevant, worker `worker-demo-a`. Note the previous state, perform one authorized action and compare the resulting state. These names illustrate the procedure; they do not identify production records or authorize creating cloud resources. For a review without changes, inspect the detail and cancel any open form.

If something fails

If a conflict appears, refresh the inventory and review current values before repeating. If the session expired, sign in and complete MFA when required. For a permission error, check the role and membership; for a provider error, inspect the operation and its safe message. Do not alter storage, revisions or controls to force

an outcome.

Related reading

- [Administrator](#)
- [Preferences](#)
- [Audit](#)

Preferences and scope

Configure the appropriate policy without mixing it with personal appearance.

Before you begin

Confirm account, tenant and active role in the account menu. Inspection requires view permission; changing records requires the specific management permission. A hidden or disabled action can reflect authorized scope; do not switch tenants to evade a denial. Work only with records in the workspace you are allowed to administer.

Procedure

1. Open Preferences and choose Notifications, Authentication, Limits, Email delivery or Data Cleanup.
2. Confirm the scope before changing values.
3. In Limits distinguish Platform defaults from tenant configuration.
4. Save only permitted settings and verify persistence.

Behavior and limits

The five categories have different authorities. The account menu provides appearance, palette and authorized membership switches; these settings do not change permissions or contract. The interfaces support day and night themes.

Access to Preferences does not authorize global authentication or SMTP changes. Button availability depends on the active role. Do not change limits to evade a commercial or security restriction.

For Notifications distinguish Preview, Test channels and Save preferences. For Authentication test the saved configuration. For Data Cleanup inspect inventory and destructive confirmation. Each action reports a different outcome and requires its own verification.

Verify the result

After saving or confirming, reopen the detail and check the state reported by the server. Distinguish an accepted request from a completed operation and retain only minimal evidence without sensitive information. When an operation is pending, wait for its final state in the corresponding view before starting another action on the same object.

Laboratory example

Use the synthetic tenant `documentation-lab` and, where relevant, worker `worker-demo-a`. Note the previous state, perform one authorized action and compare the resulting state. These names illustrate the procedure; they do not identify production records or authorize creating cloud resources. For a review without changes, inspect the detail and cancel any open form.

If something fails

If a conflict appears, refresh the inventory and review current values before repeating. If the session expired, sign in and complete MFA when required. For a permission error, check the role and membership; for a provider error, inspect the operation and its safe message. Do not alter storage, revisions or controls to force an outcome.

Related reading

- [Administrator](#)
- [Preferences](#)
- [Audit](#)

Interface language

In the user menu, select ES, EN or PT to change the interface language. The preference is saved to your profile and used by Dova and Administrator. If saving fails, the interface reports it and restores the previous choice. Available application labels change; artifact names, identities, fields, code and user data are not renamed. Style, palette and permissions retain their own configuration.

External authentication and MFA

Configure identity without weakening verification or authorization.

Before you begin

Confirm account, tenant and active role in the account menu. Inspection requires view permission; changing records requires the specific management permission. A hidden or disabled action can reflect authorized scope; do not switch tenants to evade a denial. Work only with records in the workspace you are allowed to administer.

Procedure

1. In Preferences → Authentication select Google, Microsoft, Okta, Auth0, GitHub or GitLab.
2. Register the exact Authorized callback URL with the provider and complete its fields.
3. Replace the secret only when necessary, save and test the saved configuration with an authorized identity.

Behavior and limits

External identity does not grant roles or remove MFA. New self-registration accepts verified Google identities under current policy; do not generalize this to every provider. When sign-in reports `mfaRequired`, `authenticated=false` is expected until the second factor is completed.

Expired verification requires signing in again and using the current challenge. Do not publish real codes, secrets or challenges. Preserve an authorized recovery route before changing global configuration.

Microsoft uses Tenant ID; Okta and Auth0 use Issuer URL. GitHub requires the verified primary email to match an active Dova user. Follow each card rather than generalizing fields between providers.

Verify the result

After saving or confirming, reopen the detail and check the state reported by the server. Distinguish an accepted request from a completed operation and retain only minimal evidence without sensitive information. When an operation is pending, wait for its final state in the corresponding view before starting another action on the same object.

Laboratory example

Use the synthetic tenant `documentation-lab` and, where relevant, worker `worker-demo-a`. Note the previous state, perform one authorized action and compare the resulting state. These names illustrate the procedure; they do not identify production records or authorize creating cloud resources. For a review without changes, inspect the detail and cancel any open form.

If something fails

If a conflict appears, refresh the inventory and review current values before repeating. If the session expired, sign in and complete MFA when required. For a permission error, check the role and membership; for a provider error, inspect the operation and its safe message. Do not alter storage, revisions or controls to force an outcome.

Related reading

- [Administrator](#)
- [Preferences](#)
- [Audit](#)

Administrator access

An authenticated account without Administrator permission that opens the Administrator page returns to Dova. This navigation does not grant administrative access: protected API calls still require authorization and can return 403. Without a valid session, complete normal sign-in and MFA. Use the known Dova destination; an external return URL does not expand access.

Subscription consistency during sign-in

The funding projection uses Billing's canonical period representation and is reconciled with the authorized subscription. Correcting Subscription funding projection does not match does not change the contractual period, cycle, granted credits or consumption, and does not extend a Trial. Tenant, contract and funding checks remain enforced. For a persistent error, review the existing operation and subscription before repeating registration or checkout.

Global catalog and offers

Publish versioned terms without altering previous contracts.

Before you begin

Confirm account, tenant and active role in the account menu. Inspection requires view permission; changing records requires the specific management permission. A hidden or disabled action can reflect authorized scope; do not switch tenants to evade a denial. Work only with records in the workspace you are allowed to administer.

Procedure

1. In Subscription & billing → Global catalog inspect products, plans, versions and prices.
2. Create a terms draft, review it and publish only the authorized version.
3. Use New price for its corresponding price.
4. In Tenant offers select the tenant, validity and terms, then review before publishing.

Behavior and limits

The catalog is global and an offer is bound to its tenant. Existing contracts retain their frozen terms. One hundred minor units equal one currency unit; one hundred basis points equal one percent.

Do not rewrite a used price as though it were a draft. Publishing an offer does not change another customer's offer. The current plan and contract determine actual amounts and limits; this manual is not a fixed price list.

Before publishing, confirm currency, billing interval, tax treatment and validity. A new price must reference the correct published version. Inspect the final order summary rather than a standalone amount.

Verify the result

After saving or confirming, reopen the detail and check the state reported by the server. Distinguish an accepted request from a completed operation and retain only minimal evidence without sensitive information. When an operation is pending, wait for its final state in the corresponding view before starting another action on the same object.

Laboratory example

Use the synthetic tenant `documentation-lab` and, where relevant, worker `worker-demo-a`. Note the previous state, perform one authorized action and compare the resulting state. These names illustrate the procedure; they do not identify production records or authorize creating cloud resources. For a review without changes, inspect the detail and cancel any open form.

If something fails

If a conflict appears, refresh the inventory and review current values before repeating. If the session expired, sign in and complete MFA when required. For a permission error, check the role and membership; for a provider error, inspect the operation and its safe message. Do not alter storage, revisions or controls to force an outcome.

Related reading

- [Administrator](#)
- [Preferences](#)
- [Audit](#)

Email templates, SMTP and notifications

Separate content, transport and delivery policy.

Before you begin

Confirm account, tenant and active role in the account menu. Inspection requires view permission; changing records requires the specific management permission. A hidden or disabled action can reflect authorized scope; do not switch tenants to evade a denial. Work only with records in the workspace you are allowed to administer.

Procedure

1. In Preferences → Notifications open Execution email template, select outcomes and prepare subject, preheader, HTML and plain text.
2. Preview and save.
3. In Email delivery review SMTP and sender, save and test with an authorized recipient.
4. In Notifications review events, channels, recipients, deduplication and quiet hours.

Behavior and limits

Use catalog variables such as %WORKFLOW_NAME%, %STATUS%, %RUN_ID%, %OUTPUT_ROWS% and %EXECUTION_URL%. A blank Replace password preserves the password according to the form guidance. Preview and actual receipt are separate checks.

Do not include credentials, real execution data or secret endpoints in examples. Delivery suppressed by policy is not an SMTP failure; inspect Delivery history before repeating a send.

Select at least one outcome: succeeded, failed or cancelled. Plain text is required and must communicate the same result as HTML. Quiet hours depend on their timezone and both time boundaries.

Verify the result

After saving or confirming, reopen the detail and check the state reported by the server. Distinguish an accepted request from a completed operation and retain only minimal evidence without sensitive information. When an operation is pending, wait for its final state in the corresponding view before starting another action on the same object.

Laboratory example

Use the synthetic tenant `documentation-lab` and, where relevant, worker `worker-demo-a`. Note the previous state, perform one authorized action and compare the resulting state. These names illustrate the procedure; they do not identify production records or authorize creating cloud resources. For a review without changes, inspect the detail and cancel any open form.

If something fails

If a conflict appears, refresh the inventory and review current values before repeating. If the session expired, sign in and complete MFA when required. For a permission error, check the role and membership; for a provider error, inspect the operation and its safe message. Do not alter storage, revisions or controls to force an outcome.

Related reading

- [Administrator](#)
- [Preferences](#)
- [Audit](#)

Worker installation

Prepare execution servers outside the control plane.

Before you begin

Confirm account, tenant and active role in the account menu. Inspection requires view permission; changing records requires the specific management permission. A hidden or disabled action can reflect authorized scope; do not switch tenants to evade a denial. Work only with records in the workspace you are allowed to administer.

Procedure

1. Choose agent installation on an authorized server or infrastructure provisioning through Cloud workers.
2. Download the current package for the selected system and verify integrity and prerequisites.
3. Use Register worker, define name and concurrency, generate instructions and run them on the correct server.
4. Check heartbeat, capacity and Session assignment.

Behavior and limits

Private workers support their Linux and Windows paths. Isolated public capacity is Linux-only, requires explicit enablement and approved-runtime verification. Its initial environment supports CSV and Excel file transformations, without network access or ambient cloud identity.

Installing the agent does not assign existing Sessions or transfer files. The shared sandbox exposes neither host files nor persistent checkpoints. An unsupported task fails explicitly and never falls back to an unisolated process.

The agent version can differ from 24.0.0. Use generated instructions for the selected package and system. Enter the enrollment token through the supported secure mechanism, never in public examples or shared screenshots.

Verify the result

After saving or confirming, reopen the detail and check the state reported by the server. Distinguish an accepted request from a completed operation and retain only minimal evidence without sensitive information. When an operation is pending, wait for its final state in the corresponding view before starting another action on the same object.

Laboratory example

Use the synthetic tenant `documentation-lab` and, where relevant, worker `worker-demo-a`. Note the previous state, perform one authorized action and compare the resulting state. These names illustrate the procedure; they do not identify production records or authorize creating cloud resources. For a review without changes, inspect the detail and cancel any open form.

If something fails

If a conflict appears, refresh the inventory and review current values before repeating. If the session expired, sign in and complete MFA when required. For a permission error, check the role and membership; for a provider error, inspect the operation and its safe message. Do not alter storage, revisions or controls to force an outcome.

Related reading

- [Administrator](#)
- [Preferences](#)
- [Audit](#)

Worker reassignment and retirement

Move execution references with dependency review.

Before you begin

Confirm account, tenant and active role in the account menu. Inspection requires view permission; changing records requires the specific management permission. A hidden or disabled action can reflect authorized scope; do not switch tenants to evade a denial. Work only with records in the workspace you are allowed to administer.

Procedure

1. In Cloud workers → Workers inspect the source, use Pause or Drain worker and wait for active work to finish.
2. Open Reassign references, review the authorized compatible destination and confirm impact.
3. Inspect Dependency backups to restore or reassign metadata.
4. To preserve continuity before retiring infrastructure, verify destination capacity and migrated references. To remove a VM without first migrating, follow Remove VM and unlink and review its impact.

Behavior and limits

Reassignment changes references; it does not move databases, files, operating system or credentials. A dependency backup preserves metadata and integrity; it is not a VM image or complete backup. Public retirement checks cross-tenant references and historical versions still in use.

Removal requires verified identity and ownership, permissions and a confirmed cloud result. An unknown-state read does not prove absence. Remove VM and unlink can clear references after confirmed deletion; Pause/Stop retain active-work and default-capacity protections.

Before applying, confirm that the destination satisfies the referenced objects' capabilities. Unreachable historical versions are preserved and do not block retirement by themselves. Files and connections must be available through their separate authorized procedure.

Verify the result

After saving or confirming, reopen the detail and check the state reported by the server. Distinguish an accepted request from a completed operation and retain only minimal evidence without sensitive information. When an operation is pending, wait for its final state in the corresponding view before starting another action on the same object.

Laboratory example

Use the synthetic tenant `documentation-lab` and, where relevant, worker `worker-demo-a`. Note the previous state, perform one authorized action and compare the resulting state. These names illustrate the procedure; they do not identify production records or authorize creating cloud resources. For a review without changes, inspect the detail and cancel any open form.

If something fails

If a conflict appears, refresh the inventory and review current values before repeating. If the session expired, sign in and complete MFA when required. For a permission error, check the role and membership; for a provider error, inspect the operation and its safe message. Do not alter storage, revisions or controls to force an outcome.

Related reading

- [Administrator](#)
- [Preferences](#)
- [Audit](#)

Remove a VM and recover assignments

Open Remove VM and unlink, review resource identity and impact on Sessions, executions and pools, then confirm the intended deletion. For a supported owned VM, manually dismantling every reference or first powering it off is not a prerequisite for this operation. Removing capacity can interrupt active executions; schedule the operation to suit your needs.

After confirming provider absence, Dova retires the active worker, clears Runtime references and shows Unassigned for Session bindings that used it. Definitions, published versions, configuration and audit history are preserved. If the VM is deleted directly in the provider, reconciliation applies the same treatment after verifying absence. A provider-read error is not evidence that the VM was deleted.

Before running again, select compatible authorized capacity and review the environment pool. Unassigned does not silently choose another worker. You can provision a new VM and assign it explicitly. Permissions, resource ownership, the protected base server and pending operations remain checked; removal does not bypass these protections.

ETL cleanup and lifecycle

Review destructive scope and explicitly confirm the tenant.

Before you begin

Confirm account, tenant and active role in the account menu. Inspection requires view permission; changing records requires the specific management permission. A hidden or disabled action can reflect authorized scope; do not switch tenants to evade a denial. Work only with records in the workspace you are allowed to administer.

Procedure

1. In Preferences → Data Cleanup select Tenant and Refresh inventory.
2. Review counts, blockers and the authorized backup.
3. Choose Clean tenant ETL data, compare tenant and summary, type the required confirmation and acknowledge irreversible deletion.
4. Run Delete ETL data once and inspect inventory again.

Behavior and limits

ETL cleanup includes the displayed categories of mappings, sessions, workflows, connections and credentials, runs and Repository. It preserves users, tenant, roles, billing and global administrative audit. It does not delete a VM or automatically cancel a subscription.

Active executions block the operation. A managed individual tenant may require its separate retention process. Commercial expiry never automatically deletes data; do not change type or revision to evade a blocker.

The form does not automatically generate a complete tenant backup. Preserve the authorized backup and verified recovery separately. If inventory changed since review, request it again before confirming deletion.

Verify the result

After saving or confirming, reopen the detail and check the state reported by the server. Distinguish an accepted request from a completed operation and retain only minimal evidence without sensitive information. When an operation is pending, wait for its final state in the corresponding view before starting another action on the same object.

Laboratory example

Use the synthetic tenant `documentation-lab` and, where relevant, worker `worker-demo-a`. Note the previous state, perform one authorized action and compare the resulting state. These names illustrate the procedure; they do not identify production records or authorize creating cloud resources. For a review without changes, inspect the detail and cancel any open form.

If something fails

If a conflict appears, refresh the inventory and review current values before repeating. If the session expired, sign in and complete MFA when required. For a permission error, check the role and membership; for a provider error, inspect the operation and its safe message. Do not alter storage, revisions or controls to force an outcome.

Related reading

- [Administrator](#)
- [Preferences](#)
- [Audit](#)

Connectors and authentication

Separate component capabilities, Connection access and worker availability.

Three separate responsibilities

A component defines how data is read or written. A Connection combines endpoint metadata, an authentication method and protected credential custody. A worker supplies the adapter, network access and execution capacity. A connector appearing in the catalog does not prove that a particular worker has its driver or the required resource permissions.

Dova offers delimited-file, JSON, Excel, database, API and object-storage sources; available targets are delimited file, JSON, database, API and object storage. Review the actual type in Connections and the fields offered by its form. Do not interchange authentication methods between providers merely because their names look similar.

Prepare a Connection

1. Confirm the tenant, role and permission to manage the Connection.
2. Select an offered connector and enter only authorized endpoint metadata.
3. Choose an authentication and custody method available in that form. Enter the secret through its protected control.
4. Save and inspect permitted metadata. Test the saved Connection using the available action.
5. Configure and publish a laboratory Mapping. Reference it from a Session with an eligible worker and run a small extraction from that Session.

For example, use a synthetic Connection named `CustomerLabReadOnly`, database `training` and table `customers`. These are exercise names, not instructions to connect to a real system. Verify expected columns and row counts before enabling writes. Keep the laboratory destination separate from business output.

Configure reading and output

In database table mode, selected source columns define projection. In query mode, fields depend on SQL supported by the adapter; do not assume another property rewrites arbitrary queries. Editor support does not imply that every mode works on every runtime. Review runtime compatibility before relying on a query-only feature.

File targets write their incoming schema. To exclude fields, add Field selector before the target. Successful authentication neither fixes an incorrectly configured transformation nor establishes data accuracy. Compare the actual output against a known synthetic result.

Rotate and troubleshoot

A successful connection test does not certify all reads, writes, retry policies or results. If the test passes but execution fails, inspect the worker, network, resource permissions, published version and specific operation. Check dependencies before rotating or revoking a credential.

Dova Vault protects custody; it does not authorize exposing secrets or internal references. Avoid pasting credentials into SQL, descriptions, parameters or conversations. Access to public worker capacity does not grant Connection access or make data and Sessions public. Verify the final result and retain only authorized evidence.

Mapping component reference

Purpose, settings and limits of all 36 available components.

Using this reference

The catalogue contains 36 components: 6 Sources, 17 Transform, 8 Quality and 5 Targets. Visible names match Dova; identifiers and JSON keys are not translated. A component available in the editor does not establish that a connection is configured or that every worker can execute it.

Each example is a partial, synthetic configuration. Complete required fields for your case, authorized connections and runtime bindings before validation. Never place passwords inside the Mapping.

Sources

Flat File input (`source_csv`)

Reads delimited CSV, TSV or TXT. Headers determine columns; `headerMappings` projects, renames and converts only mapped fields. Without a header, `columns` assigns positional names.

Settings: `format`, `delimiter`, `hasHeader`, `headerRowNumber`, `columns`, `headerMappings`, `encoding`.

`format`: `csv`, `tsv`, `txt`; `encoding`: `utf-8`, `utf-8-bom`.

Synthetic example:

```
{
  "format": "csv",
  "hasHeader": true,
  "delimiter": ",",
}
```

Copy

Limits: Use a one-character delimiter and UTF-8. With `hasHeader=true`, `columns` is not a field-selection switch. Set the header row before reviewing data.

JSON input (`source_json`)

Reads objects, JSON arrays or JSON Lines as records. `recordPath` locates records; `fieldMappings` selects source paths and output names/types. Without mappings it retains observed top-level fields and serializes nested objects or arrays as JSON text.

Settings: `format`, `recordPath`, `fieldMappings`, `encoding`.

`format`: `json`, `jsonl`; `encoding`: `utf-8`, `utf-8-bom`.

Synthetic example:

```
{
  "format": "json",
  "recordPath": "items"
}
```

Copy

Limits: Paths use dotted names, not general JSONPath expressions. A missing nonnullable mapped field causes an error.

Excel input (`source_excel`)

Reads workbook sheets and ranges. Select sheets, headers and row/column bounds; `headerMappings` projects, renames and converts fields. `includeSheetName` can add sheet provenance.

Settings: `workbookFormat`, `sheetMode`, `sheetName`, `sheetNames`, `sheetPattern`, `sheetNameRegex`, `includeHiddenSheets`, `hasHeader`, `headerRowNumber`, `dataStartRowNumber`, `startColumn`, `endColumn`, `columns`, `headerMappings`, `skipEmptyRows`, `includeSheetName`, `sheetNameField`, `passwordProtected`.

`workbookFormat`: `auto`, `xlsx`, `xlsm`, `xls`; `sheetMode`: `single`, `list`, `pattern`, `regex`, `all`.

Synthetic example:

```
{
  "sheetMode": "single",
  "sheetName": "Sheet1",
  "hasHeader": true,
  "headerRowNumber": 1,
  "dataStartRowNumber": 2
}
```

Copy

Limits: Reader formats are `xlsx`, `xlsm` and `xls`. Sheets and ranges must exist. Supply a workbook password only for the authorized operation; do not store it in Mapping configuration.

Database input (`source_database`)

Reads a table/view or a read-only query through a compatible worker. In `table` mode, `selectedColumns` projects fields and `schema / objectName` identify the relation. In `query` mode, output columns come from `sqlQuery`.

Settings: `connectionId`, `sourceMode`, `schema`, `objectType`, `objectName`, `sqlQuery`, `selectedColumns`, `fetchSize`, `queryTimeoutSeconds`.

`sourceMode`: `table`, `query`; `objectType`: `table`, `view`.

Synthetic example:

```
{
  "sourceMode": "table",
  "objectName": "customers",
  "selectedColumns": [
    "name",
    "amount"
  ]
}
```

Copy

Limits: `selectedColumns` does not rewrite arbitrary SQL. Generic connector streaming supports table/view mode; queries depend on the connector and runtime. Selecting a connection type does not configure credentials.

API input (`source_api`)

Reads records with `GET` through a configured connection. `responsePath` locates records and `fieldMappings` projects nested paths into typed logical fields. Pagination can be `none`, `page`, `offset` or `cursor`.

Settings: `connectionId`, `method`, `resourcePath`, `responsePath`, `fieldMappings`, `paginationMode`, `pageSize`, `retries`, `timeoutSeconds`.

`method`: `GET`; `paginationMode`: `none`, `page`, `offset`, `cursor`.

Synthetic example:

```
{
  "method": "GET",
  "resourcePath": "/customers",
  "paginationMode": "none"
}
```

Copy

Limits: Actual support depends on the provider. SaaS connectors use their own resources/entities; they do not all accept generic REST pagination options.

Object storage input (`source_object`)

Reads CSV or JSONL from object storage. `resource` identifies the object, `selectedColumns` projects fields and `delimiter` applies to CSV.

Settings: `connectionId`, `resource`, `format`, `selectedColumns`, `delimiter`.

`format`: `csv`, `jsonl`.

Synthetic example:

```
{
  "resource": "input/customers.csv",
  "format": "csv",
  "selectedColumns": [
    "name",
    "amount"
  ],
  "delimiter": ",",
}
```

Copy

Limits: Requires a compatible connection, read permissions and a stable tabular schema. Displaying the component does not establish storage access.

Transform

Row filter (`filter`)

Keeps rows matching one field condition and preserves all their columns and values.

Settings: `field`, `operator`, `value`.

`operator`: `equals`, `not_equals`, `contains`, `starts_with`, `ends_with`, `gt`, `gte`, `lt`, `lte`, `not_empty`.

Synthetic example:

```
{
  "field": "status",
  "operator": "equals",
  "value": "active"
}
```

Copy

Limits: `equals` and `not_equals` compare case-sensitive text. `contains`, `starts_with` and `ends_with` ignore case. `gt/gte/lt/lte` convert to numbers. `not_empty` excludes null and blank text.

Field selector (`select`)

Projects each row and its output schema onto only the configured existing fields, in configured order. Drops all other fields while keeping row count.

Settings: `fields`.

Synthetic example:

```
{
  "fields": [
    "First_name",
    "Last_name"
  ]
}
```

Copy

Limits: Does not transform values, rename fields or filter rows. At least one selected field must exist. Unexpected final fields require checking the executed revision, connected branch and downstream components.

Field mapper (`rename`)

Renames fields listed in `mapping` while preserving their values, unmapped fields and all rows.

Settings: `mapping`.

Synthetic example:

```
{
  "mapping": {
    "old_name": "new_name"
  }
}
```

Copy

Limits: Output names must remain unique. It does not replace values or drop unmapped fields; add Field selector when projection is needed.

Calculated field (derive)

Adds or replaces one field using literal text and `${field}` or `${param.name}` interpolation. Copies all other fields and rows.

Settings: `field`, `template`.

Synthetic example:

```
{
  "field": "label",
  "template": "ID-${id}"
}
```

Copy

Limits: It is not an expression engine: no arithmetic, `LEFT`, `SUBSTR`, slicing or initial capitalization. Output is text bounded to 4,000 characters.

Convert type (convert)

Converts a field to `string`, `integer`, `number`, `boolean`, `date` or `datetime`. An empty `outputField` replaces the source; a named output keeps the source and writes the converted result there.

Settings: `field`, `targetType`, `outputField`, `onError`, `defaultValue`.

`targetType`: `string`, `integer`, `number`, `boolean`, `date`, `datetime`; `onError`: `fail`, `null`, `keep`, `default`.

Synthetic example:

```
{
  "field": "amount",
  "targetType": "number",
  "outputField": "amount_number",
  "onError": "fail"
}
```

Copy

Limits: `date` requires `YYYY-MM-DD`; `datetime` uses ISO 8601. There is no custom date format. `onError` accepts `fail`, `null`, `keep` or `default`; numbers must be finite.

Clean text (clean)

Applies operations in the configured order only to `fields`. Preserves all rows and columns; unselected fields keep their values and types, and null stays null.

Settings: `fields`, `operations`. Type at least one letter in Operations to see available matches and choose an operation.

Operation	Effect
<code>trim</code>	Removes whitespace from both ends.
<code>ltrim</code>	Removes whitespace from the left end.

Operation	Effect
<code>rtrim</code>	Removes whitespace from the right end.
<code>collapse_whitespace</code>	Reduces whitespace groups to one space.
<code>lower</code>	Converts text to lowercase.
<code>upper</code>	Converts text to uppercase.
<code>capitalize</code>	Uppercases the first character and lowercases the remainder.
<code>title</code>	Capitalizes each word; whitespace and punctuation separate words.
<code>remove_accents</code>	Removes accent marks.
<code>remove_line_breaks</code>	Replaces each group of line breaks with one space, preserving word separation.

To transform `jORGE` into `Jorge`, configure:

```
{
  "fields": ["first_name"],
  "operations": ["trim", "capitalize"]
}
```

Copy

Order matters: `trim` removes surrounding whitespace before capitalization. Selected nonnull values become text; avoid `capitalize` when the remainder of a name must retain its original case.

Replace values (`replace`)

Replaces literal text in one field while preserving other columns and all rows. It is the direct choice for changing exactly `jorge` to `Jorge`.

Settings: `field`, `find`, `replacement`, `matchMode`, `caseSensitive`.

`matchMode`: `contains`, `exact`, `starts_with`, `ends_with`.

Synthetic example:

```
{
  "field": "name",
  "find": "jorge",
  "replacement": "Jorge",
  "matchMode": "exact",
  "caseSensitive": true
}
```

Copy

Limits: It does not use regular expressions. `matchMode` accepts `exact`, `contains`, `starts_with`, `ends_with`; `caseSensitive` controls case matching. This specific replacement does not capitalize arbitrary names.

Split field (split)

Splits text by a literal delimiter into `outputFields` . `keepSource` controls retaining the original field; `trimParts` trims pieces.

Settings: `field` , `delimiter` , `outputFields` , `keepSource` , `trimParts` .

Synthetic example:

```
{
  "field": "full_name",
  "delimiter": " ",
  "outputFields": [
    "first_name",
    "last_name"
  ],
  "keepSource": true,
  "trimParts": true
}
```

Copy

Limits: It does not extract characters by position. Splitting occurs at most one fewer time than the output count, leaving the remainder in the last field. Missing pieces become empty strings.

Concatenate fields (concat)

Joins values as text into `outputField` using `separator` ; preserves source fields and all rows.

Settings: `fields` , `outputField` , `separator` , `skipEmpty` .

Synthetic example:

```
{
  "fields": [
    "first_name",
    "last_name"
  ],
  "outputField": "full_name",
  "separator": " ",
  "skipEmpty": true
}
```

Copy

Limits: It does not add numbers. `skipEmpty` skips blank or whitespace-only pieces; null becomes empty text.

Join (join)

Combines exactly two immediate inputs on equality keys. Supports `inner` , `left` , `right` , `full` ; emits both schemas and uses `rightPrefix` for right-side name collisions.

Settings: `joinType` , `leftInputId` , `rightInputId` , `leftKey` , `rightKey` , `conditions` , `keyType` , `strategy` , `caseSensitive` , `trimStrings` , `nullsEqual` , `rightPrefix` .

`joinType` : `inner` , `left` , `right` , `full` ; `keyType` : `auto` , `string` , `integer` , `number` , `boolean` , `date` , `datetime` ; `strategy` : `auto` , `hash` , `sorted` .

Synthetic example:

```
{
  "joinType": "left",
  "leftInputId": "customers",
  "rightInputId": "orders",
  "leftKey": "id",
  "rightKey": "customer_id",
  "keyType": "auto",
  "strategy": "auto",
  "caseSensitive": true,
  "trimStrings": true,
  "nullsEqual": false,
  "rightPrefix": "right_"
}
```

Copy

Limits: Set valid `leftInputId` , `rightInputId` and keys. Multiple conditions use AND. Many-to-many matches multiply rows. Nulls do not match by default. Expansion limits and `auto` , `hash` , `sorted` strategies apply.

Limit rows (`limit`)

Keeps a contiguous window of rows after `offset` , preserving columns and values.

Settings: `count` , `offset` .

Synthetic example:

```
{
  "count": 10,
  "offset": 0
}
```

Copy

Limits: `count` : 1–250,000; `offset` : 0–250,000. It does not sort: add Row sort first for a top-N by a field.

Row sort (`sort`)

Sorts rows by one field in `asc` or `desc` direction, preserving all columns and rows.

Settings: `field` , `direction` .

`direction` : `asc` , `desc` .

Synthetic example:

```
{
  "field": "amount",
  "direction": "desc"
}
```

Copy

Limits: Numerically convertible values sort numerically; other text ignores case. This is not multi-key sorting. Streaming requires a staging step.

Group aggregate (aggregate)

Produces one row per `groupBy` group, containing only grouping fields and the metric output. `count` counts rows; `sum` and `avg` aggregate one field.

Settings: `groupBy`, `metric`, `field`, `outputField`.

`metric`: `count`, `sum`, `avg`.

Synthetic example:

```
{
  "groupBy": [
    "region"
  ],
  "metric": "sum",
  "field": "amount",
  "outputField": "total"
}
```

Copy

Limits: It does not calculate between columns in the same row and has no `min / max`. `sum / avg` ignore values that cannot become finite numbers. Metric output is text. Empty `groupBy` groups all nonempty input.

Unpivot (unpivot)

Turns each column in `fields` into a row: `keyField` receives its original name and `valueField` its value. Keeps `keepFields`; when empty, retains the other input fields.

Settings: `fields`, `keepFields`, `keyField`, `valueField`, `dropNulls`.

Synthetic example:

```
{
  "fields": [
    "jan",
    "feb"
  ],
  "keepFields": [
    "id"
  ],
  "keyField": "month",
  "valueField": "amount",
  "dropNulls": false
}
```

Copy

Limits: Expands rows. Output names cannot overwrite existing columns. `dropNulls` removes only null, keeping zero, `false` and empty text. This is not pivot.

Run sequence (sequence)

Adds an integer sequence `start + row index × step`, preserving rows, fields and order.

Settings: `outputField`, `start`, `step`.

Synthetic example:

```
{
  "outputField": "sequence",
  "start": 1,
  "step": 1
}
```

Copy

Limits: Restarts each run; it is not a persistent or globally unique identity. `step` cannot be zero and output cannot overwrite an existing field. Safe integer bounds apply.

Field fingerprint (`hash_fields`)

Adds a SHA-256 hexadecimal fingerprint over selected field names and canonical typed JSON values. Preserves original data.

Settings: `fields`, `outputField`.

Synthetic example:

```
{
  "fields": [
    "id",
    "name"
  ],
  "outputField": "fingerprint"
}
```

Copy

Limits: Null and empty text differ. This is neither encryption nor anonymization. Input fields must exist and output cannot overwrite them; JSON depth is limited to 20 and numbers must be finite.

Partition rank (`rank`)

Adds a numeric position within `partitionBy` groups while retaining original row order. `dense` shares consecutive ranks; `rank` leaves gaps; `row_number` assigns unique positions.

Settings: `partitionBy`, `field`, `direction`, `mode`, `outputField`.

`direction`: `asc`, `desc`; `mode`: `dense`, `rank`, `row_number`.

Synthetic example:

```
{
  "partitionBy": [
    "region"
  ],
  "field": "amount",
  "direction": "desc",
  "mode": "dense",
  "outputField": "rank"
}
```

Copy

Limits: The ordered field must be finite numeric on every row; null, blank and boolean fail. Output cannot overwrite existing columns. Empty `partitionBy` treats all input as one group. Final output is not sorted.

Quality

Default value (`defaults`)

Fills a field when null, blank or both according to `when` ; preserves present values, other columns and all rows. It may add a missing field.

Settings: `field` , `value` , `when` .

`when` : `null_or_empty` , `null` , `empty` .

Synthetic example:

```
{
  "field": "name",
  "value": "Unknown",
  "when": "null_or_empty"
}
```

Copy

Limits: Zero and `false` are retained. It does not replace nonempty literal values; use Replace values for that goal.

Deduplicate (`dedupe`)

Keeps the first row of each key combination with all its fields. Empty `fields` compares all input columns.

Settings: `fields` .

Synthetic example:

```
{
  "fields": [
    "customer_id"
  ]
}
```

Copy

Limits: Drops duplicates but does not merge values or automatically choose the latest record. Sort first if the retained first row matters. Comparison uses typed JSON values.

Quality check (`validate`)

Checks one field for empty values. Fails execution when their percentage exceeds `maxNullPercent` ; otherwise passes every row and field unchanged.

Settings: `field` , `maxNullPercent` .

Synthetic example:

```
{
  "field": "name",
  "maxNullPercent": 0
}
```

[Copy](#)

Limits: Null and whitespace-only text are empty; zero and `false` are not. The threshold is 0–100. It does not fill or remove values.

Required fields (`quality_required`)

Marks a row invalid when any required field is null or blank. Execution fails if the percentage exceeds `maxInvalidPercent`.

Settings: `fields`, `maxInvalidPercent`.

Synthetic example:

```
{
  "fields": [
    "id",
    "name"
  ],
  "maxInvalidPercent": 0
}
```

[Copy](#)

Limits: On threshold success, even invalid rows remain. It does not fill data or discard records.

Type check (`quality_type`)

Checks whether values can convert to `expectedType` without converting them. Preserves rows and values when the threshold is not exceeded.

Settings: `field`, `expectedType`, `allowEmpty`, `maxInvalidPercent`.

`expectedType`: `string`, `integer`, `number`, `boolean`, `date`, `datetime`.

Synthetic example:

```
{
  "field": "amount",
  "expectedType": "number",
  "allowEmpty": false,
  "maxInvalidPercent": 0
}
```

[Copy](#)

Limits: Use `Convert` type for transformation. `allowEmpty` controls missing values. Numeric text may pass and remain text.

Pattern check (`quality_pattern`)

Checks full-field matching against a safe regular expression. Preserves data when `maxInvalidPercent` is not exceeded.

Settings: `field`, `pattern`, `allowEmpty`, `maxInvalidPercent`.

Synthetic example:

```
{
  "field": "code",
  "pattern": "[A-Z]{3}",
  "allowEmpty": false,
  "maxInvalidPercent": 0
}
```

Copy

Limits: Safety limits apply to `pattern`. It does not extract, replace or filter values. `allowEmpty` controls missing values separately.

Range check (`quality_range`)

Checks finite numbers against `minimum` and/or `maximum`, with inclusive or exclusive bounds. Keeps data on threshold success.

Settings: `field`, `minimum`, `maximum`, `inclusive`, `allowEmpty`, `maxInvalidPercent`.

Synthetic example:

```
{
  "field": "age",
  "minimum": "18",
  "maximum": "100",
  "inclusive": true,
  "allowEmpty": false,
  "maxInvalidPercent": 0
}
```

Copy

Limits: At least one bound is required. It does not clamp or replace out-of-range values. `allowEmpty` controls missing values.

Allowed values (`quality_domain`)

Checks exact textual membership in `allowedValues`, respecting `caseSensitive`. Preserves data on threshold success.

Settings: `field`, `allowedValues`, `caseSensitive`, `allowEmpty`, `maxInvalidPercent`.

Synthetic example:

```
{
  "field": "status",
  "allowedValues": [
    "active",
    "inactive"
  ],
  "caseSensitive": true,
  "allowEmpty": false,
}
```

Copy

```
"maxInvalidPercent": 0
}
```

Limits: The list cannot be empty. It does not map or replace values; use Replace values to transform them. `allowEmpty` controls missing values.

Targets

Flat File output (`target_csv`)

Writes rows as CSV or TXT using upstream output columns and their order. Configure header, delimiter, quoting, nulls, encoding and line endings.

Settings: `format`, `fileName`, `delimiter`, `includeHeader`, `textQualifier`, `quoteMode`, `escapeCharacter`, `lineEnding`, `encoding`, `nullValue`.

```
format: csv, txt; textQualifier: double, single, none; quoteMode: minimal, all;
lineEnding: lf, crlf; encoding: utf-8, utf-8-bom.
```

Synthetic example:

```
{
  "fileName": "output.csv",
  "format": "csv",
  "delimiter": ",",
  "includeHeader": true
}
```

Copy

Limits: It has no `fields / columns` selector. Use Field selector before the target to remove fields. `fileName` controls output naming, not projection.

JSON output (`target_json`)

Writes a JSON array or JSON Lines using only upstream output columns. `includeNulls=false` omits null-valued fields.

Settings: `format`, `fileName`, `pretty`, `includeNulls`, `lineEnding`, `encoding`.

```
format: json, jsonl; lineEnding: lf, crlf; encoding: utf-8, utf-8-bom.
```

Synthetic example:

```
{
  "fileName": "output.jsonl",
  "format": "jsonl",
  "includeNulls": true
}
```

Copy

Limits: It does not select arbitrary fields: use Field selector. `pretty` controls JSON array formatting.

Database output (`target_database`)

Writes upstream fields to a table through a compatible worker. `append` adds rows; `replace` and `upsert` depend on destination capabilities. `keyColumns` defines upsert keys.

Settings: `connectionId`, `schema`, `objectName`, `writeMode`, `batchSize`, `commitIntervalRows`, `keyColumns`, `createIfMissing`, `queryTimeoutSeconds`.

`writeMode`: `append`, `replace`, `upsert`.

Synthetic example:

```
{
  "objectName": "customers",
  "writeMode": "append",
  "batchSize": 5000
}
```

Copy

Limits: `replace` can replace existing data. Do not assume universal support for `createIfMissing`, write modes or query timeouts. Verify retries and keys before production.

API output (`target_api`)

Sends records with `POST`, `PUT` or `PATCH`. Supports `append / upsert` with success codes, batches, retries and timeout where supported by the REST runtime.

Settings: `connectionId`, `method`, `resourcePath`, `writeMode`, `keyColumns`, `uniqueKeysEnforced`, `batchSize`, `successCodes`, `retries`, `timeoutSeconds`.

`method`: `POST`, `PUT`, `PATCH`; `writeMode`: `append`, `upsert`.

Synthetic example:

```
{
  "method": "POST",
  "resourcePath": "/customers",
  "writeMode": "append",
  "batchSize": 100
}
```

Copy

Limits: Upsert requires exactly one key. Idempotence depends on the API. `uniqueKeysEnforced` describes provider enforcement; it does not create a uniqueness constraint.

Object storage output (`target_object`)

Writes upstream fields as CSV/JSONL to object storage using conditional replacement. Requires `idempotencyKey` and exactly one precondition.

Settings: `connectionId`, `resource`, `format`, `writeMode`, `delimiter`, `idempotencyKey`, `ifAbsent`, `expectedEtag`, `expectedGeneration`.

`format`: `csv`, `jsonl`; `writeMode`: `replace`.

Synthetic example:

```
{
  "resource": "output/customers.csv",
  "format": "csv",
  "writeMode": "replace",
  "idempotencyKey": "example-run",
  "ifAbsent": true
}
```

[Copy](#)

Limits: Use `ifAbsent` , `expectedEtag` or `expectedGeneration` , never several. With ETag/generation set `ifAbsent=false` . There is no unconditional append mode; verify connection and permissions.

Execution compatibility

Generic streaming requires one linear path with a compatible source and target. Join and multi-input graphs need an execution path supporting that topology; do not change the algorithm merely to hide a compatibility error. Row sort, Group aggregate, Partition rank and quality gates may stage data before emitting results. Streaming does not mean unlimited memory or immediate output for every component.

Verify the result

Validate the graph, save and publish deliberately. Run a synthetic sample through authorized runtime and compare schema, rows, ordering and values. If CSV has extra fields after Field selector, check its saved settings, the Mapping revision used by the Session and downstream components: the selector does remove fields from both schema and rows.

Transformations and quality checks

Choose components by their actual effect on values, columns and rows.

Choose by the intended effect

Use Row filter to remove rows, Field selector to remove columns and Field mapper to rename them. To change the exact value `jorge` to `Jorge`, use Replace values with `field=name`, `find=jorge`, `replacement=Jorge`, `matchMode=exact` and `caseSensitive=true`. No additional function is needed when that rule is already defined.

For name cleanup, use Clean text with `trim`, `capitalize` or `title`, according to the expected result. `capitalize` uppercases the first character and lowercases the remainder; `title` capitalizes each word. Calculated field interpolates text: `LEFT(name,1)` does not execute a function.

Quality contract

Quality includes Default value and Deduplicate, which modify data. The other six checks are gates: Quality check, Required fields, Type check, Pattern check, Range check and Allowed values. If invalid percentage exceeds the threshold, the whole execution fails. Otherwise every row passes, including invalid ones. They do not produce a rejected-row output. With zero threshold, one invalid row in nonempty input causes failure.

Transformations and quality

Row filter (`filter`)

Keeps rows matching one field condition and preserves all their columns and values.

Settings: `field`, `operator`, `value`.

`operator`: `equals`, `not_equals`, `contains`, `starts_with`, `ends_with`, `gt`, `gte`, `lt`, `lte`, `not_empty`.

Synthetic example:

```
{
  "field": "status",
  "operator": "equals",
  "value": "active"
}
```

Copy

Limits: `equals` and `not_equals` compare case-sensitive text. `contains`, `starts_with` and `ends_with` ignore case. `gt/gte/lt/lte` convert to numbers. `not_empty` excludes null and blank text.

Field selector (`select`)

Projects each row and its output schema onto only the configured existing fields, in configured order. Drops all other fields while keeping row count.

Settings: `fields`.

Synthetic example:

```
{
  "fields": [
    "First_name",
    "Last_name"
  ]
}
```

Copy

Limits: Does not transform values, rename fields or filter rows. At least one selected field must exist. Unexpected final fields require checking the executed revision, connected branch and downstream components.

Field mapper (`rename`)

Renames fields listed in `mapping` while preserving their values, unmapped fields and all rows.

Settings: `mapping`.

Synthetic example:

```
{
  "mapping": {
    "old_name": "new_name"
  }
}
```

Copy

Limits: Output names must remain unique. It does not replace values or drop unmapped fields; add Field selector when projection is needed.

Calculated field (`derive`)

Adds or replaces one field using literal text and `${field}` or `${param.name}` interpolation. Copies all other fields and rows.

Settings: `field`, `template`.

Synthetic example:

```
{
  "field": "label",
  "template": "ID-${id}"
}
```

Copy

Limits: It is not an expression engine: no arithmetic, `LEFT`, `SUBSTR`, slicing or initial capitalization. Output is text bounded to 4,000 characters.

Convert type (`convert`)

Converts a field to `string`, `integer`, `number`, `boolean`, `date` or `datetime`. An empty `outputField` replaces the source; a named output keeps the source and writes the converted result there.

Settings: `field`, `targetType`, `outputField`, `onError`, `defaultValue`.

targetType: string, integer, number, boolean, date, datetime; onError: fail, null, keep, default.

Synthetic example:

```
{
  "field": "amount",
  "targetType": "number",
  "outputField": "amount_number",
  "onError": "fail"
}
```

Copy

Limits: date requires YYYY-MM-DD; datetime uses ISO 8601. There is no custom date format. onError accepts fail, null, keep or default; numbers must be finite.

Clean text (clean)

Applies operations in the configured order only to fields. Preserves all rows and columns; unselected fields keep their values and types, and null stays null.

Settings: fields, operations. Type at least one letter in Operations to see available matches and choose an operation.

Operation	Effect
trim	Removes whitespace from both ends.
ltrim	Removes whitespace from the left end.
rtrim	Removes whitespace from the right end.
collapse_whitespace	Reduces whitespace groups to one space.
lower	Converts text to lowercase.
upper	Converts text to uppercase.
capitalize	Uppercases the first character and lowercases the remainder.
title	Capitalizes each word; whitespace and punctuation separate words.
remove_accents	Removes accent marks.
remove_line_breaks	Replaces each group of line breaks with one space, preserving word separation.

To transform jORGE into Jorge, configure:

```
{
  "fields": ["first_name"],
  "operations": ["trim", "capitalize"]
}
```

Copy

Order matters: `trim` removes surrounding whitespace before capitalization. Selected nonnull values become text; avoid `capitalize` when the remainder of a name must retain its original case.

Replace values (`replace`)

Replaces literal text in one field while preserving other columns and all rows. It is the direct choice for changing exactly `jorge` to `Jorge`.

Settings: `field`, `find`, `replacement`, `matchMode`, `caseSensitive`.

`matchMode`: `contains`, `exact`, `starts_with`, `ends_with`.

Synthetic example:

```
{
  "field": "name",
  "find": "jorge",
  "replacement": "Jorge",
  "matchMode": "exact",
  "caseSensitive": true
}
```

Copy

Limits: It does not use regular expressions. `matchMode` accepts `exact`, `contains`, `starts_with`, `ends_with`; `caseSensitive` controls case matching. This specific replacement does not capitalize arbitrary names.

Split field (`split`)

Splits text by a literal delimiter into `outputFields`. `keepSource` controls retaining the original field; `trimParts` trims pieces.

Settings: `field`, `delimiter`, `outputFields`, `keepSource`, `trimParts`.

Synthetic example:

```
{
  "field": "full_name",
  "delimiter": " ",
  "outputFields": [
    "first_name",
    "last_name"
  ],
  "keepSource": true,
  "trimParts": true
}
```

Copy

Limits: It does not extract characters by position. Splitting occurs at most one fewer time than the output count, leaving the remainder in the last field. Missing pieces become empty strings.

Concatenate fields (`concat`)

Joins values as text into `outputField` using `separator`; preserves source fields and all rows.

Settings: `fields`, `outputField`, `separator`, `skipEmpty`.

Synthetic example:

```
{
  "fields": [
    "first_name",
    "last_name"
  ],
  "outputField": "full_name",
  "separator": " ",
  "skipEmpty": true
}
```

Copy

Limits: It does not add numbers. `skipEmpty` skips blank or whitespace-only pieces; null becomes empty text.

Join (join)

Combines exactly two immediate inputs on equality keys. Supports `inner`, `left`, `right`, `full`; emits both schemas and uses `rightPrefix` for right-side name collisions.

Settings: `joinType`, `leftInputId`, `rightInputId`, `leftKey`, `rightKey`, `conditions`, `keyType`, `strategy`, `caseSensitive`, `trimStrings`, `nullsEqual`, `rightPrefix`.

`joinType`: `inner`, `left`, `right`, `full`; `keyType`: `auto`, `string`, `integer`, `number`, `boolean`, `date`, `datetime`; `strategy`: `auto`, `hash`, `sorted`.

Synthetic example:

```
{
  "joinType": "left",
  "leftInputId": "customers",
  "rightInputId": "orders",
  "leftKey": "id",
  "rightKey": "customer_id",
  "keyType": "auto",
  "strategy": "auto",
  "caseSensitive": true,
  "trimStrings": true,
  "nullsEqual": false,
  "rightPrefix": "right_"
}
```

Copy

Limits: Set valid `leftInputId`, `rightInputId` and keys. Multiple conditions use AND. Many-to-many matches multiply rows. Nulls do not match by default. Expansion limits and `auto`, `hash`, `sorted` strategies apply.

Default value (defaults)

Fills a field when null, blank or both according to `when`; preserves present values, other columns and all rows. It may add a missing field.

Settings: `field`, `value`, `when`.

`when`: `null_or_empty`, `null`, `empty`.

Synthetic example:

```
{
  "field": "name",
  "value": "Unknown",
  "when": "null_or_empty"
}
```

Copy

Limits: Zero and `false` are retained. It does not replace nonempty literal values; use Replace values for that goal.

Limit rows (`limit`)

Keeps a contiguous window of rows after `offset` , preserving columns and values.

Settings: `count` , `offset` .

Synthetic example:

```
{
  "count": 10,
  "offset": 0
}
```

Copy

Limits: `count` : 1–250,000; `offset` : 0–250,000. It does not sort: add Row sort first for a top-N by a field.

Deduplicate (`dedupe`)

Keeps the first row of each key combination with all its fields. Empty `fields` compares all input columns.

Settings: `fields` .

Synthetic example:

```
{
  "fields": [
    "customer_id"
  ]
}
```

Copy

Limits: Drops duplicates but does not merge values or automatically choose the latest record. Sort first if the retained first row matters. Comparison uses typed JSON values.

Row sort (`sort`)

Sorts rows by one field in `asc` or `desc` direction, preserving all columns and rows.

Settings: `field` , `direction` .

`direction` : `asc` , `desc` .

Synthetic example:

```
{
  "field": "amount",
  "direction": "desc"
}
```

[Copy](#)

Limits: Numerically convertible values sort numerically; other text ignores case. This is not multi-key sorting. Streaming requires a staging step.

Group aggregate (aggregate)

Produces one row per `groupBy` group, containing only grouping fields and the metric output. `count` counts rows; `sum` and `avg` aggregate one field.

Settings: `groupBy`, `metric`, `field`, `outputField`.

`metric`: `count`, `sum`, `avg`.

Synthetic example:

```
{
  "groupBy": [
    "region"
  ],
  "metric": "sum",
  "field": "amount",
  "outputField": "total"
}
```

[Copy](#)

Limits: It does not calculate between columns in the same row and has no `min / max`. `sum / avg` ignore values that cannot become finite numbers. Metric output is text. Empty `groupBy` groups all nonempty input.

Unpivot (unpivot)

Turns each column in `fields` into a row: `keyField` receives its original name and `valueField` its value. Keeps `keepFields`; when empty, retains the other input fields.

Settings: `fields`, `keepFields`, `keyField`, `valueField`, `dropNulls`.

Synthetic example:

```
{
  "fields": [
    "jan",
    "feb"
  ],
  "keepFields": [
    "id"
  ],
  "keyField": "month",
  "valueField": "amount",
  "dropNulls": false
}
```

[Copy](#)

Limits: Expands rows. Output names cannot overwrite existing columns. `dropNulls` removes only null, keeping zero, `false` and empty text. This is not pivot.

Run sequence (`sequence`)

Adds an integer sequence `start + row index × step`, preserving rows, fields and order.

Settings: `outputField`, `start`, `step`.

Synthetic example:

```
{
  "outputField": "sequence",
  "start": 1,
  "step": 1
}
```

Copy

Limits: Restarts each run; it is not a persistent or globally unique identity. `step` cannot be zero and output cannot overwrite an existing field. Safe integer bounds apply.

Field fingerprint (`hash_fields`)

Adds a SHA-256 hexadecimal fingerprint over selected field names and canonical typed JSON values. Preserves original data.

Settings: `fields`, `outputField`.

Synthetic example:

```
{
  "fields": [
    "id",
    "name"
  ],
  "outputField": "fingerprint"
}
```

Copy

Limits: Null and empty text differ. This is neither encryption nor anonymization. Input fields must exist and output cannot overwrite them; JSON depth is limited to 20 and numbers must be finite.

Partition rank (`rank`)

Adds a numeric position within `partitionBy` groups while retaining original row order. `dense` shares consecutive ranks; `rank` leaves gaps; `row_number` assigns unique positions.

Settings: `partitionBy`, `field`, `direction`, `mode`, `outputField`.

`direction`: `asc`, `desc`; `mode`: `dense`, `rank`, `row_number`.

Synthetic example:

```
{
  "partitionBy": [
```

Copy

```

    "region"
  ],
  "field": "amount",
  "direction": "desc",
  "mode": "dense",
  "outputField": "rank"
}

```

Limits: The ordered field must be finite numeric on every row; null, blank and boolean fail. Output cannot overwrite existing columns. Empty `partitionBy` treats all input as one group. Final output is not sorted.

Quality check (`validate`)

Checks one field for empty values. Fails execution when their percentage exceeds `maxNullPercent` ; otherwise passes every row and field unchanged.

Settings: `field` , `maxNullPercent` .

Synthetic example:

```

{
  "field": "name",
  "maxNullPercent": 0
}

```

Copy

Limits: Null and whitespace-only text are empty; zero and `false` are not. The threshold is 0-100. It does not fill or remove values.

Required fields (`quality_required`)

Marks a row invalid when any required field is null or blank. Execution fails if the percentage exceeds `maxInvalidPercent` .

Settings: `fields` , `maxInvalidPercent` .

Synthetic example:

```

{
  "fields": [
    "id",
    "name"
  ],
  "maxInvalidPercent": 0
}

```

Copy

Limits: On threshold success, even invalid rows remain. It does not fill data or discard records.

Type check (`quality_type`)

Checks whether values can convert to `expectedType` without converting them. Preserves rows and values when the threshold is not exceeded.

Settings: `field` , `expectedType` , `allowEmpty` , `maxInvalidPercent` .

`expectedType`: `string`, `integer`, `number`, `boolean`, `date`, `datetime`.

Synthetic example:

```
{
  "field": "amount",
  "expectedType": "number",
  "allowEmpty": false,
  "maxInvalidPercent": 0
}
```

Copy

Limits: Use `Convert` type for transformation. `allowEmpty` controls missing values. Numeric text may pass and remain text.

Pattern check (`quality_pattern`)

Checks full-field matching against a safe regular expression. Preserves data when `maxInvalidPercent` is not exceeded.

Settings: `field`, `pattern`, `allowEmpty`, `maxInvalidPercent`.

Synthetic example:

```
{
  "field": "code",
  "pattern": "[A-Z]{3}",
  "allowEmpty": false,
  "maxInvalidPercent": 0
}
```

Copy

Limits: Safety limits apply to `pattern`. It does not extract, replace or filter values. `allowEmpty` controls missing values separately.

Range check (`quality_range`)

Checks finite numbers against `minimum` and/or `maximum`, with inclusive or exclusive bounds. Keeps data on threshold success.

Settings: `field`, `minimum`, `maximum`, `inclusive`, `allowEmpty`, `maxInvalidPercent`.

Synthetic example:

```
{
  "field": "age",
  "minimum": "18",
  "maximum": "100",
  "inclusive": true,
  "allowEmpty": false,
  "maxInvalidPercent": 0
}
```

Copy

Limits: At least one bound is required. It does not clamp or replace out-of-range values. `allowEmpty` controls missing values.

Allowed values (`quality_domain`)

Checks exact textual membership in `allowedValues` , respecting `caseSensitive` . Preserves data on threshold success.

Settings: `field` , `allowedValues` , `caseSensitive` , `allowEmpty` , `maxInvalidPercent` .

Synthetic example:

```
{
  "field": "status",
  "allowedValues": [
    "active",
    "inactive"
  ],
  "caseSensitive": true,
  "allowEmpty": false,
  "maxInvalidPercent": 0
}
```

Copy

Limits: The list cannot be empty. It does not map or replace values; use `Replace values` to transform them. `allowEmpty` controls missing values.

Execution compatibility

Generic streaming requires one linear path with a compatible source and target. Join and multi-input graphs need an execution path supporting that topology; do not change the algorithm merely to hide a compatibility error. Row sort, Group aggregate, Partition rank and quality gates may stage data before emitting results. Streaming does not mean unlimited memory or immediate output for every component.

Verify the result

Validate the graph, save and publish deliberately. Run a synthetic sample through authorized runtime and compare schema, rows, ordering and values. If CSV has extra fields after Field selector, check its saved settings, the Mapping revision used by the Session and downstream components: the selector does remove fields from both schema and rows.

Parameters, templates and conditions

Distinguish typed values, text interpolation and Workflow decisions.

Before you start

Typed parameters, Calculated field templates and Workflow JSON conditions are different mechanisms. None turns arbitrary text into executable code. Define types and scope before adding overrides. Use synthetic values and never place secrets in text parameters.

Supported types are `string`, `integer`, `number`, `boolean`, `datetime`, `json` and `secret`. A sensitive parameter must use `secret` with a valid protected reference, rather than cleartext marked sensitive.

Procedure

1. Declare a parameter with a unique name, type and constraints. Up to 100 definitions are allowed. Names start with a letter, have at most 64 characters and allow letters, numbers, dots, hyphens and underscores. `sys.` and `dova.` are reserved.
2. Review `required`, `overridable`, `scopes`, default and constraints. Boolean requires JSON `true / false`, not equivalent-looking text. Numbers must be finite.
3. Check layer resolution: default → global → environment → worker → task. A later permitted layer replaces an earlier one without overriding scope restrictions.
4. For derived text use `${name}` and `${param.region}`. Output is text up to 4,000 characters; a recognized missing identifier becomes empty text.
5. For branching, use a JSON condition with authorized references, not a SQL formula.

Synthetic example

Calculated field template:

```
{
  "field": "label",
  "template": "${name}-${param.region}"
}
```

Copy

With `name=Ana` and parameter `region=NORTH`, output is `Ana-NORTH`. It does not add numbers or extract a first letter.

A Decision True condition:

```
{
  "ref": "parameters.RUN_MODE",
  "op": "eq",
  "value": "full"
}
```

Copy

Declare `RUN_MODE` before using it. Run separate attempts with `full` and `sample`; review the complementary False branch shown in the interface.

Verify the result

Condition references support `parameters.name`, `variables.name`, `tasks.id.status` and `tasks.id.outputs.name`. Use actual task identifiers and documented outputs; do not invent a universal row counter. Operators are `eq`, `ne`, `gt`, `gte`, `lt`, `lte`, `in`, `not_in`, `contains`, `exists`, `truthy`. Groups `all` / `any` combine 1–20 expressions; `not` negates one expression.

Confirm values and provenance in authorized preview. Then check the selected branch in Monitor. Incompatible comparison types do not promise automatic conversion.

Troubleshooting

If an override has no effect, inspect scope, Overridable and source layer. For empty template output, check the field or parameter name. Dotted names do not provide general navigation through nested objects. Condition trees reject depth over five, more than 40 nodes and comparison values over 4,096 bytes. Missing references make ordinary comparisons false. Correct the contract rather than weakening restrictions to accept an unauthorized value.

Runtime operational variables

Open Sessions → Session → Runtime → Available runtime variables for available tokens, meanings, examples and the last authorized resolved value. `%RUN_TIMESTAMP%` contributes to the output filename when Target filename timestamp is effectively Enabled. The form example is a preview; the actual value belongs to an attempt ID and date. These operational variables do not replace `${field}` or permit arbitrary code. Sensitive parameters remain masked.

Permissions and actions

Understand why viewing a module, editing an object and executing it require different checks.

Authorization context

Identity, tenant membership and the selected role determine the working context. A Repository location does not replace those checks. An active account may have an inactive membership; a visible role does not automatically grant every permission. The server checks current authority for each action and may reject an operation that was previously available.

Do not confuse Administrator access with the platform's unique master identity. The master user's DovaAI quota exemption requires the persisted identity, active membership and selected platform role. It is not granted to every administrator or by matching an email address. Provider readiness and authorization controls still apply.

Review an action

Goal	What to check
Read	Tenant, membership and module or object read permission
Edit	Change permission, current version and editable state
Publish	Saved draft, dependencies and publication permission
Execute	Execution permission, plan, runtime, capacity and Connections
Delete or purge	Scope, references, blockers and current confirmation

Authorization does not guarantee available resources. A valid plan does not grant missing permissions either. Being able to view a Session does not mean it can be executed, and publishing a Workflow does not itself activate a schedule. Check the specific operation rather than inferring privileges from a module name.

Investigate denied access

1. Inspect your identity and tenant in the current account context.
2. Confirm the selected role and active membership.
3. Identify the actual action: reading, editing, publishing, executing or deleting.
4. Ask an administrator to review that authorization and scope without indiscriminately expanding privileges.
5. Refresh the view after an accepted change and check the resource again.

For a synthetic exercise, someone who only needs to inspect Monitor should review authorized attempts and results, not receive Repository purge permission. Delegate the required action rather than a larger permission set intended to hide a configuration problem.

Tenant and execution boundaries

A public worker shares eligible capacity; it does not share other tenants' objects, conversations or data. Connections retain their own restrictions. DovaAI reads and recommends within authorized scope. It never creates, edits, publishes, runs or deletes artifacts, even when requested or confirmed; a chat instruction cannot grant new permissions.

If identity, tenant or permissions change during an operation, the context must be checked again. Do not reuse an old confirmation or treat a hidden button as the sole security control. The server response and accepted object state determine the outcome. Keep authorized audit evidence without copying credentials or another tenant's information.

Environment context

Before using this module, confirm Tenant, Domain and Environment in the user menu. The initial environment is Development. Module, Domain and environment capabilities apply together; Production is read only without explicit authorization. Switching the selector does not move artifacts or publish or execute their data. See [environments and recovery](#) and [environment promotion](#).

Environment permissions

Authorization combines module permissions, Domain access and environment capabilities. The current catalog contains 23 environment permissions: `etl.environments.view`, `etl.environments.{development, test, stage, production}.{view, edit, publish, run, runtime.manage}`, `etl.environments.promote` and `etl.environments.runtime.manage`. Global view and Runtime grants retain their explicit scope; a grant for one environment does not grant access to all others. Module management does not replace these capabilities.

ETL Developer can work in Development, Test and Stage and has no Production access by default. Release Manager can review all four environments and publish into Test, Stage and Production for authorized promotions; this does not grant Production editing or execution. Promotion retains preview, exact published dependencies and explicit Production approval. Custom roles can grant the required capabilities per environment. A user identity is not duplicated per environment; check the active role, membership and effective permissions.

Production Mapping, Session and Workflow editors remain read only: accepted versions arrive through promotion. Configuring existing or new Production Connections is a separate capability requiring `etl.connections.manage` and `etl.environments.production.edit`; it does not unlock those three editors or legacy read-only Connections.

Dova 23.0.2: capabilities and acceptance

Review functional changes and verify behavior in your authorized environment.

Scope of this edition

This manual describes Dova 23.0.2. Confirm your installation's active version before following a procedure. This reference retains its historical route to preserve existing links; the title and edition identify its current content.

Monitor and operational diagnosis

Compact logs combine timestamp, level, component and message with filters, following and display pause. Cards and live progress read counters from the current attempt; phases, rows, bytes and rate depend on evidence reported by the worker. Unknown totals are not represented as completion percentages. The list updates its cells while retaining filters, selection and scroll.

The Gantt uses compact rows and distinguishes Workflow, Session and Mapping through kind and hierarchy. Session duration belongs within its orchestration rather than additional time to add to the Workflow bar. Open the attempt to inspect the versions actually executed.

Safe worker errors retain classified causes, phases and available limits. An input exceeding a profile maximum differs from a protected existing destination. Free physical disk does not increase input limits or executor temporary memory. Earlier attempts without original exceptions do not acquire retrospective evidence.

Session and dependencies

Sessions → Runtime shows available variables, meanings, examples and the last actual value associated with a run. `%RUN_TIMESTAMP%` contributes to output filenames when timestamps are effectively enabled. Inherit with a disabled global preference does not apply the token even when written in the template. A preview illustrates settings rather than proving an actual output filename.

The compact Workflow chain retains name, kind, consumed version and newer-version availability. Update versions changes the draft; Save draft → Publish adopts that revision for later attempts. Reload status and opening an artifact only inspect it.

Read-only DovaAI

DovaAI follows the user's language and recommends the right component, fields and values through brief steps. To replace exactly jorge with Jorge it recommends Replace values with exact matching; Field selector retains only chosen fields. It uses the public manual and authorized Workflow and Session evidence to separate confirmed causes from hypotheses.

The assistant never creates, edits, configures, publishes, runs or deletes artifacts, even with confirmation. You make changes through the normal controls. This policy replaces assisted artifact creation from earlier versions.

Environment promotions

Configuration → Promotions follows Development → Test → Stage → Production. It promotes an immutable bundle containing published Workflow, Sessions and Mappings and their exact versions, applies destination bindings and creates separate artifacts without changing the originals. Review the diff, worker, plan, DPU and concurrency before confirming. Production requires explicit approval and a comment.

History supports manifest export with Git provenance and restoration of historical definitions through new versions. It contains no credentials or usable secret references. Promotion does not execute Workflows, enable schedules, undo external data or start GitHub Actions. Development, Test and Stage use Development processing; Production uses the effective plan's Production processing.

Acceptance

1. Confirm version, tenant, permissions and location.
2. Inspect profiles, variables and dependencies before editing settings.
3. Use a laboratory and synthetic data to test an identifiable output.
4. Compare logs, counters, versions, schema and the final file.
5. Personally review and confirm each promotion or restoration; retain its evidence.

Changes in 23.0.1

1. Compact Timestamp and Message logs with expandable structured details.
2. Optional Workflow and Session TRACE with inheritance and generation disabled by default.
3. Gantt typography and tooltips aligned with Dova.
4. Compact List with W/S badges, hierarchy and aligned columns.
5. Existing-row and new-execution updates without flicker, preserving reading context.
6. Delete removes the selection when focus is in the Mappings canvas and outside text editors.
7. Clean text suggests operations from the first letter and adds `capitalize`, `title`, `ltrim`, `rtrim` and `remove_line_breaks`.
8. Sessions → General orders Domain → Folder → Mapping and loads choices on opening.
9. DovaAI recommends the component directly, a short flow and steps with exact properties and values; it remains strictly read only.

Changes in 23.0.2

1. Monitor → List and Gantt display the WF and SES icons and badges used in workspace tabs.
2. Full Screen enters and leaves the Mappings and Workflows canvas in the same editor, preserving selection, zoom, position and DovaAI access.
3. DOVA_DATA_GRID actions show progress and block other actions in the same grid until completion or failure.
4. Filters use states applicable to their module; Mappings offers Draft and Published.
5. Mappings adds Used by (Sessions), Location, Last published and Description, configurable in Columns.
6. Administrator grids preserve menus and position during updates. Cloud Workers provides native controls according to provider and an ordered preparation flow to resolve dependencies and deprovision in the cloud before removing the Dova registration.

7. Administrator reuses frequent reads in RAM with expiry, limits and invalidation, respecting tenant, role and permissions.
8. Overview updates charts without unmounting them, checks changes every five seconds and identifies the last confirmed read when refresh fails.
9. Overview permanently includes `vm-cloudata-pe` as the Dova Base server for platform administrators, separate from execution workers and visible even without metrics.

This edition retains Linux worker 1.3.14. VM actions respect actual provider capabilities: native Pause and Resume are available for GCP; Azure offers Start, Stop and Deprovision; AWS VM lifecycle management is unavailable in this edition. States and results are checked against their source; refreshing a screen does not complete an operation.

Environments: isolate Development, Test and Production

Select Tenant, Domain and Environment while keeping each context's work separate.

Select the context

The Tenant, Domain and Environment summary appears in the bottom operational status bar of Dova and Administrator. Click the summary to open the user menu with all three selectors. It retains the environment badge and color without a top strip or duplicate fields.

The user menu combines Tenant, Domain and Environment. The initial context is Development. Canonical environments are Development, Test, Stage and Production; Stage appears according to configuration and permissions. A Domain/Folder location organizes ownership and access, while the environment separates definitions and executions. Sharing a Domain does not grant access to another environment.

1. Confirm the authorized Tenant.
2. Select the Domain in the user menu.
3. Choose Environment and wait for context loading to finish.
4. Check the DEV, TEST, STAGE or PROD badge before editing or running.

Mappings, Sessions, Workflows, Connections and Monitor read the selected context. The editor's Domain identifies the owner and is locked; Folder belongs to that owner. Use the explicit Move operation and its permissions to change location.

Save, publish and run

Save draft in DEV saves a Development draft. Publish in DEV publishes that version in the same environment. Promote to Test opens a separate destination review. Labels reflect the current environment; saving or publishing neither moves a definition between environments nor executes its data.

Authorization combines module permission, Domain access and environment and artifact capabilities. Production is read only unless explicitly authorized. An operator can have run permission without edit permission. Environment names do not grant capabilities.

Switch without mixing drafts

Tabs, filters, selection, drafts and positions are retained for each Tenant, Domain and Environment. A switch cancels previous reads and discards stale responses. The context cannot change while a write is in flight. Recovery verifies the artifact and its authorization before restoring a dirty editor.

Changing environment preserves local work without saving or publishing it to the server. Password fields and selected files are not transferred to another context. Save or close a dirty Session before changing Tenant. When the server cannot confirm the context, the platform becomes read only and offers Retry context.

Earlier definitions and recovery

The upgrade does not automatically assign all earlier artifacts to Development. Shared dependencies and published versions can remain through verified membership and exact references. An ambiguous or shared artifact can be read only; earlier local drafts remain pending recovery.

Use Fork to Development to create an editable copy. Its review includes indirect Mapping, Session or Workflow dependencies. Bind each source Connection to a compatible Connection already owned by Development. Dova does not copy credentials. The server rechecks the revision and rejects source or destination changes since preview. The copy is a draft; Open in Development is a separate explicit action. Review and publish it before execution.

See [the promotion guide](#) for the forward path to Test and Production.

Environment permissions

Authorization combines module permissions, Domain access and environment capabilities. The current catalog contains 23 environment permissions: `etl.environments.view`, `etl.environments.{development,test,stage,production}.{view,edit,publish,run,runtime.manage}`, `etl.environments.promote` and `etl.environments.runtime.manage`. Global view and Runtime grants retain their explicit scope; a grant for one environment does not grant access to all others. Module management does not replace these capabilities.

ETL Developer can work in Development, Test and Stage and has no Production access by default. Release Manager can review all four environments and publish into Test, Stage and Production for authorized promotions; this does not grant Production editing or execution. Promotion retains preview, exact published dependencies and explicit Production approval. Custom roles can grant the required capabilities per environment. A user identity is not duplicated per environment; check the active role, membership and effective permissions.

Production Mapping, Session and Workflow editors remain read only: accepted versions arrive through promotion. Configuring existing or new Production Connections is a separate capability requiring `etl.connections.manage` and `etl.environments.production.edit`; it does not unlock those three editors or legacy read-only Connections.

Public Trial: platform-managed Development pool

An eligible public Trial without an explicit Development policy can automatically use a platform-managed public Development pool. It includes only verified public shared workers authorized for that Trial; private workers and workers assigned to other configured pools are excluded. It neither saves a Runtime policy nor grants permission to manage one.

An explicit Development policy, including an empty policy, takes precedence over this automatic option. Test, Stage and Production retain their policies. Capacity remains subject to availability, profiles, quotas and concurrency; eligibility does not guarantee an available worker.

GitOps kit per workspace

The [GitOps kit 1.1.2](#) binds Tenant, Domain and Environment for queries, draft reservations, publications and leases. Use `source.environment` and `source.domainId` for exports; `target.environment` and `bindings.domainId` for imports. Defaults are Development and General; preserve state per target. See the [Finance example and state recovery](#). Direct Production authoring requires governed promotion rather than the import command.

Before stopping or retiring capacity

Pause VM and Stop VM retain active-work, default-worker and shared-capacity checks; review references and impact before using them. Pause scheduling stops new assignments and lets current work finish. To retire the VM permanently, use Remove VM and unlink and review its confirmation: after verifying cloud absence, Dova clears assignments and shows Unassigned without deleting definitions or published versions. Assign compatible capacity explicitly before running again.

Actions require the module permission and the relevant environment's Runtime capability, or an explicitly granted global permission. A shared VM requires authorization across all its environments.

Promote a release between environments

Review dependencies, destination Connections, permissions and approval before applying.

Prepare the source

A promotion starts from a published Mapping, Session or Workflow version. Select the source Tenant, Domain and Environment. Save and publish any changes to include; a local draft is outside the reviewed bundle.

Open Promote to Test from the editor or Configuration → Environment promotions. Select artifact type, published version and destination. The usual forward path is Development → Test → Production; Stage is optional when available. Copying from Production back to Development uses Fork to Development, a separate operation.

Review the bundle and its bindings

1. Inspect the preview and exact published dependency chain. A Workflow includes the Sessions and Mappings it actually references; a Session includes its Mapping.
2. Assign existing destination Connections. Use authorized, compatible identifiers; credentials remain in Dova Vault.
3. Review applicable worker, execution profile, parameters and input/output bindings. The worker must be permitted in the destination environment's pool.
4. Check the effective plan, available DPU, concurrency and processing limits.
5. Inspect differences, versions and Git provenance before continuing.

After changing version, configuration or context, generate a new preview. The server rechecks permissions, revisions and configuration during apply; an old review does not authorize a changed package.

Confirm and verify

Apply only the reviewed bundle. Production requires explicit approval and a comment, together with the necessary permissions. The operation creates separate destination definitions while preserving source versions. It does not run the pipeline or enable its schedules automatically.

In history, confirm environment, version, actor and result. Open the destination, inspect Connections and Runtime and perform an authorized manual run against test data and a test destination. In Monitor, verify consumed versions, rows and the final file before enabling a schedule.

Restore a version

Rollback is a manual operation on an earlier promoted version and its saved configuration. Review the current destination revision and confirm again; Production retains its approval requirement. Restoring definitions does not undo writes to files, databases or other external services.

Git and DevOps

Record repository, commit or tag provenance and export the manifest of an applied promotion. Dova's promotion API supports preview and apply with authorized authentication and permissions. GitOps kit 1.1.2 exports the manifest through `promotion-export`; its separate definition-plan/apply flow imports only into Development, Test or Stage. Keep the manifest and digest in your review process.

This integration does not perform Git push, provider OAuth, webhooks or automatic deployments by itself. An external pipeline must configure its own permissions and deliberately invoke available operations. DovaAI can explain a failure and recommend steps, but never approves, applies, restores, configures or executes a promotion.

See [environments and recovery](#) and [CLI and API](#).

Prepare destination Connections

Before promoting to Production, prepare target-owned Connections with valid custody and permissions. An authorized administrator can register them or change metadata with `etl.connections.manage` and `etl.environments.production.edit`. This specific authorization does not unlock Production Mapping, Session or Workflow editors. Shared or ambiguous legacy Connections stay read only.

Promotion and Fork to Development previews offer only compatible existing destination Connections. Select each replacement explicitly; passwords are not copied and secret references are not fabricated. Connection Domain and Environment remain tied to their ownership context.

GitOps kit per workspace

The [GitOps kit 1.1.2](#) binds Tenant, Domain and Environment for queries, draft reservations, publications and leases. Use `source.environment` and `source.domainId` for exports; `target.environment` and `bindings.domainId` for imports. Defaults are Development and General; preserve state per target. See the [Finance example and state recovery](#). Direct Production authoring requires governed promotion rather than the import command.

Dova 24, 25 and 26: scope and changes

Dova 26.0.0 changes, preserving the functional baseline and historical Dova 24 and 25 references.

Current edition

This edition describes Dova 26.0.0. Check your installation's active version before following a procedure. This reference keeps the route used by the Dova 24 manual; its historical sections remain below.

Changes in 26.0.0

1. DovaAI lets you select an element or area and review a screenshot and its safe context before sending them.
2. The launcher follows mouse or touch dragging, previews its docking space and animates docking and detaching from the bottom bar. The reduced-motion preference is respected.
3. Mappings Configuration organizes component fields, help and actions within the panel; Header mapping retains its own scrolling for columns.
4. The bottom bar prioritizes Tenant, Domain, Environment and connection state. Operational details groups Active, Workers, Queue, Plan, Autosave and Updated; space remains available for docked DovaAI.
5. Advanced, Validate, Save draft and Publish in the Mappings editor use icons with help and accessible names. Run is removed: the Session supplies execution and requires compatible capacity.
6. The Mappings header no longer repeats DEV; the environment stays visible in the bottom operational status bar.
7. The Mappings list removes Environment and Runtime filters and uses the active context.
8. The Mappings list aligns Search, the remaining filters and icon actions.
9. Sessions and Workflows lists adopt the same action toolbar. They remove only Environment and retain Runtime and the other filters.

Saving, publishing, promoting and running remain separate actions subject to permissions. This documentation describes controls and procedures; verify the active version and each operation's outcome in your installation.

Functional baseline in 25.0.0

1. Tenant, Domain and Environment are selected from the user menu; a compact indicator keeps context and its colors visible.
2. Environment-specific permissions and Release Manager separate development from promotion to Production. ETL Developer has no Production access by default.
3. Administrator scopes workers, cloud connections, profiles, operations and pools to authorized environments; users and roles keep a shared identity.
4. The Administrator sidebar removes circular counters and follows Dova's visual and navigation foundation.
5. Module loading and a bounded RAM read cache reduce repeated requests while respecting identity, role and context.
6. Configuration → Miscellaneous customizes the shared Tenant+Environment color, with preview and restore-default controls.
7. ES, EN and PT select interface language from the user menu.

8. VM controls use implemented provider capabilities and actual state and periodically reconcile cloud changes.
9. Remove VM and unlink retires owned capacity and, after cloud absence is confirmed, clears assignments and projects Unassigned without deleting versions or audit history.
10. The disk chart identifies free space, capacity, usage and sample time using Dova's visual standard.
11. CPU, memory and disk show Not available without current telemetry; historical samples remain distinct.

These capabilities retain permissions, ownership, pending-operation checks and base-server protection. Each action depends on the implemented provider and effective authorization.

Changes in 25.0.1

1. Compact VM controls follow real provider state, with Start for Stopped and Resume for Paused according to permissions.
2. A DovaAI icon movable with mouse, touch and keyboard, with a local position per authenticated user.
3. Element or area selection shares context and a real screenshot prepared with redaction.
4. Compact chat supports pasted or dropped images, preview and removal before sending.
5. Dova and Administrator share left navigation and a user menu beside its button, fitted to the screen.
6. The menu's Domain list follows create, update and delete operations while preserving the unsaved-change guard.

Fixes in 25.0.2

1. Administrator HTML navigation redirects an authenticated account without administrative permission to Dova; API authorization remains enforced.
2. Funding-period reconciliation avoids Subscription funding projection does not match through Billing's canonical projection, without changing the period, cycle or granted credits.
3. Repository loads context with expired temporary drafts and offers recovery after read errors instead of remaining indefinitely in Loading.

The Tenant, Domain and Environment summary appears in the bottom operational status bar of Dova and Administrator. Click the summary to open the user menu with all three selectors. It retains the environment badge and color without a top strip or duplicate fields.

See this manual's DovaAI, account-menu, Cloud workers and Repository procedures. VM operations, writes and publications require their own permissions and result verification.

Historical Dova 24.0.0 reference

Edition and active version

This manual describes Dova 24.0.0. Check your installation's active version before following a procedure. Documentation describes capabilities; it does not establish that an installation's promotion, execution or publication completed.

Main changes

1. Global Tenant, Domain and Environment context, initially Development.

2. Mappings, Sessions, Workflows, Connections and Monitor catalogs separated by context and authorized by the server.
3. Tabs, filters and local work retained per context; stale responses discarded and switching blocked during writes.
4. Save and publish controls identify their environment; Production requires explicit capabilities to modify definitions.
5. Fork to Development recovers definitions as separate drafts with compatible destination Connections and dependency review, without copying credentials.
6. Mapping, Session or Workflow promotions with exact published dependencies, preview, destination bindings, Production approval and manual restoration history.
7. Environment worker policies and distinct edit, publish, run, promote and Runtime administration capabilities.
8. DovaAI receives environment context and drops earlier conversations on a context switch. It remains a read-only diagnostic assistant.

Upgrade of earlier artifacts

Earlier artifacts without clear ownership are not automatically assigned to Development. Published and shared references are preserved through verified membership and an exact dependency chain. An ambiguous definition may need review and Fork to Development. Switching the selector never retags an artifact's owner environment or updates published versions.

Earlier custom roles retain reading according to their permissions; new environment mutation capabilities require an explicit grant. Custom environments are never an implicit alias for Development or Production.

Limits and responsibility

Promotion and definition rollback neither execute pipelines nor reverse external data. Destination Connections keep their own credential custody. Worker availability, plan quotas and successful execution are checked separately.

DovaAI identifies causes and recommends options; users alone change configuration and confirm operations. Git provenance and manifest export support external review, without granting Git access or starting a DevOps pipeline by themselves.

See [environments](#) and [promotion](#). The [historical reference](#) keeps its route and earlier-edition content.

GitOps kit per workspace

The [GitOps kit 1.1.2](#) binds Tenant, Domain and Environment for queries, draft reservations, publications and leases. Use `source.environment` and `source.domainId` for exports; `target.environment` and `bindings.domainId` for imports. Defaults are Development and General; preserve state per target. See the [Finance example and state recovery](#). Direct Production authoring requires governed promotion rather than the import command.