Digital Contracting Service

User Manual

As of 2026-08-05

User Manual: Digital Contracting Service (DCS)

What the DCS is

The Digital Contracting Service (DCS) accompanies contracts through their entire lifecycle: from reusable contract templates through contract creation and negotiation with a partner organization to signing, activation, monitoring, archiving and auditing.

A contract in the DCS always has two sides. People sign the **readable contract document**. The **machine-readable contract content** can be evaluated, checked and handed over to a target system by the system itself. Both come from the same source, so they cannot drift apart. Every step (creating, changing, offering, approving, signing) is recorded in an audit-proof way and remains traceable later.

Sign-in works without username and password: you identify yourself with your digital wallet. The credentials you present there determine your roles and with them everything you can see and do in the system.

How this manual is organized

Every task in the DCS belongs to a role, so this manual is organized by role. Read the chapter for your role. The cross-references take you to the neighbouring steps.

Chapter	Role	Content
Login and Navigation	all	Signing in with the wallet, home page, side navigation, access protection
Creating Templates	Template Creator	Designing components and contract templates, clauses with machine-readable rules
Reviewing Templates	Template Reviewer	Reviewing submitted templates and forwarding them for approval
Approving Templates	Template Approver	Giving reviewed templates final approval or rejecting them
Managing Templates	Template Manager	Registering templates, template catalogue, Semantic Hub
Creating and Negotiating Contracts	Contract Creator	Deriving contracts from templates, filling them in, offering, negotiating
Reviewing Contracts	Contract Reviewer	Reviewing negotiated contracts and forwarding them
Approving Contracts	Contract Approver	Releasing reviewed contracts for signing
Signing Contracts	Contract Signer	Guided signing in the Secure Contract Viewer with your own wallet
Managing Contracts	Contract Manager	Contract overview, target system, activation, signature validation, termination
Running Audits	Auditor	Justified, recorded audits across contracts, templates, signatures and the archive
Managing the Archive and Retrieving Contracts	Archive Manager	Archive overview, retrieving archived contracts, erasure of contract content
Monitoring Compliance	Compliance Officer	Risk monitoring, incident reporting, signature audit history
System Administration	Sys. Administrator	Target systems, system users and their credentials, key inventory
Troubleshooting and FAQ	all	Error messages, edge cases and known limitations

User roles

Which areas a person sees and may operate is determined solely by the roles carried in their wallet credentials. There is no role management in the application.

Around templates

- **Template Creator:** designs reusable components and contract templates and submits them for review.
- **Template Reviewer:** reviews submitted templates and forwards them for approval or rejects them.
- **Template Approver:** gives reviewed templates final approval.

- **Template Manager:** registers approved templates in the template catalogue and maintains the shared vocabulary in the Semantic Hub.

Around contracts

- **Contract Creator:** derives contracts from registered templates, fills them in, offers them to the counterparty, negotiates and can withdraw an offer.
- **Contract Negotiator:** negotiates existing contracts; sees the same areas as the Contract Creator, without contract creation.
- **Contract Reviewer:** reviews negotiated contracts before approval.
- **Contract Approver:** releases reviewed contracts for signing.
- **Contract Signer:** signs approved contracts with their own wallet and their own signing key.
- **Contract Manager:** keeps track of the contract portfolio, sets the target system, activates contracts, validates signatures, terminates contracts and creates renewal contracts (Renew).
- **Contract Observer:** reads contracts and signatures without triggering any actions.

Organization-wide

- **Auditor:** runs justified, recorded audits across templates, contracts, signatures and archive operations and exports audit reports.
- **Archive Manager:** oversees the contract archive, retrieves archived contracts and carries out erasure requests.
- **Compliance Officer:** monitors the system for rule violations and reports incidents as formal reports.
- **Sys. Administrator:** registers target systems, manages system users and their credentials and inspects the key inventory.
- **Integration Manager:** manages the target systems for connecting adjacent systems.

A person can hold several roles; the navigation then shows the union of the corresponding areas.

The contract lifecycle in brief

Templates and contracts pass through fixed stations, shown in the views as a progress bar:

- **Templates:** Draft → Submitted → Reviewed → Approved → Registered (in use)
- **Contracts:** Draft → Submitted → Reviewed → Approved → Signed → Active

A contract can also take on further states that the progress bar does not show as separate stations: **OFFERED** (offered to the counterparty), **NEGOTIATION** (under negotiation), **REJECTED** and **WITHDRAWN** (rejected or withdrawn) and, after signing, **REVOKED**, **TERMINATED** and **EXPIRED**. In the lists they appear as status badges.

A status only changes through the actions of the responsible role (**Submit**, **Approve**, **Register**, **Offer to counterparty**, **Accept offer**, **Withdraw**, **Deploy**, **Renew**, **Terminate**). There is no editable status field.

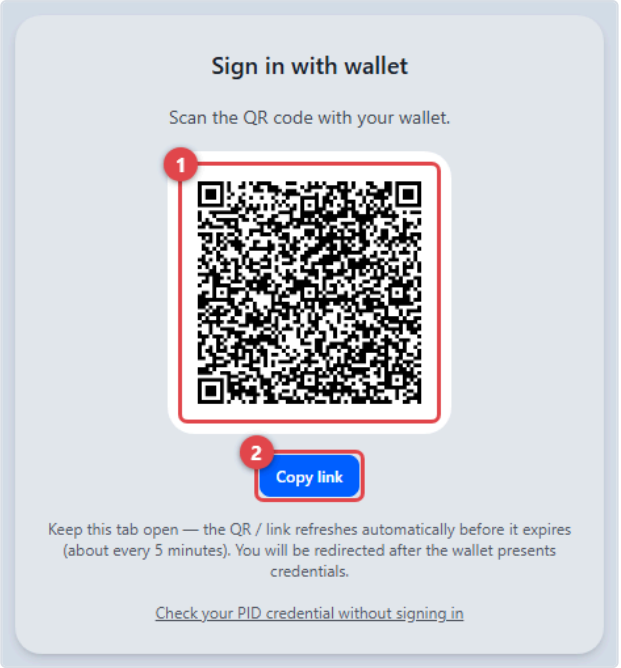
The five core flows at a glance

Task	Role	Chapter
Create a contract	Contract Creator	Creating and Negotiating Contracts
Review a contract	Contract Reviewer	Reviewing Contracts
Approve a contract	Contract Approver	Approving Contracts
Sign a contract	Contract Signer	Signing Contracts
Retrieve a contract **from the archive**	Archive Manager, Auditor	Managing the Archive and Retrieving Contracts

Login and Navigation

Signing in with the wallet

The DCS has no usernames and passwords. You sign in with your digital wallet.



When you open the application, the **Sign in with wallet** page appears.

1. Scan the QR code **(1)** with your wallet. Alternatively, **Copy link (2)** puts the same sign-in link on the clipboard if you want to transfer it to the wallet by hand.
2. Present your credentials in the wallet: your identity and your roles. You are then redirected into the application automatically.

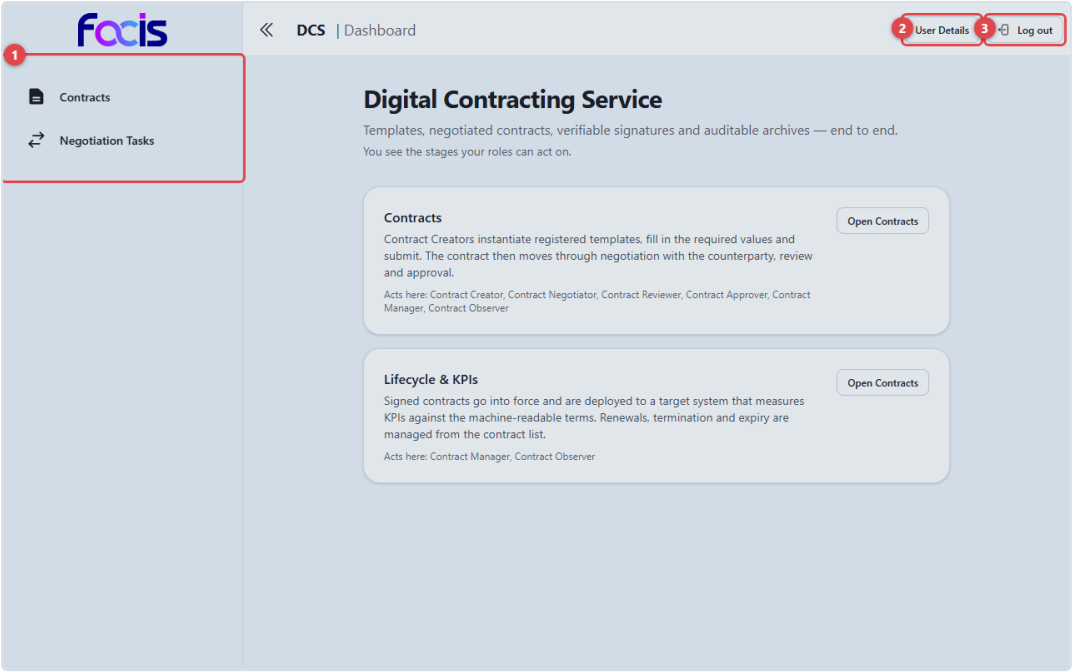
The note below the code explains that the QR code and the link renew themselves as long as the page stays open (roughly every five minutes). Keep the tab open until the redirect has happened.

At the very bottom, the link **Check your PID credential without signing in** leads to a separate page where you can present and check your identity credential without signing in. This helps when sign-in fails and you want to know whether the credential is the cause.

Which areas you see afterwards is determined solely by the roles in the presented credentials. There is no role setting in the application.

Home page

After signing in you land on the home page.



The home page explains the stations of the contract lifecycle in short sections and shows for each station which roles act there ("Acts here: ..."). A button leads directly into the corresponding work area (for example **Open Contracts**). You only see the stations where your own roles take part.

1. The side navigation **(1)** on the left contains only the areas of your roles, in the example those of a Contract Creator with **Contracts** and **Negotiation Tasks**.
2. **User Details (2)** shows your signed-in identity and your roles.
3. **Log out (3)** signs you out.

The double arrow at the top left of the header collapses and expands the sidebar. When collapsed it shows only the icons; the name of an entry then appears as a tooltip at the mouse pointer.

Side navigation

The left sidebar is the central entry point. Depending on your roles, the following entries appear there:

Entry	Visible to
Templates	Template Creator, Template Reviewer, Template Approver, Template Manager
Contracts	Contract Creator, Contract Negotiator, Contract Reviewer, Contract Approver, Contract Manager, Contract Observer
Review Tasks	Template Reviewer, Template Manager, Contract Reviewer
Approval Tasks	Template Approver, Template Manager, Contract Approver
Negotiation Tasks	Contract Creator, Contract Negotiator, Contract Reviewer
Template Catalogue	Template Manager
Audit	Auditor, Archive Manager
Archive	Auditor, Archive Manager
Signing	Contract Signer, Contract Manager, Contract Observer
Compliance Viewer	Auditor, Compliance Officer, Contract Manager
Non-Compliance Investigation	Compliance Officer
Semantic Hub	Template Manager
Target Systems	Sys. Administrator, Integration Manager
System Users	Sys. Administrator
Key Inventory	Sys. Administrator

Working with lists

The overviews for templates, contracts and tasks share the same layout:

- A **search field** with a selector in front of it. The selector determines what is searched; the default is the unique identifier (**DID**). The other options are **Name**, **Description**, **Version** and **Contract Data** or **Template Data**. **Search** runs the search. In the task lists the search only matches the **Name**; the selector has no effect there.
- **Filter** narrows the list by status (for example only **DRAFT** or only **APPROVED**). Clicking the same status again removes the restriction.
- **Sort by** changes the sort field and direction.
- Below the list you set the page size and page through the results.

Access protection

If you open a view for which you lack the role, the application redirects you back to the home page without any further message. In addition, the server checks every single action. A rejected call appears as a red notice, for example "insufficient permissions: requires one of [...]". Seeing a control on screen does not by itself grant permission.

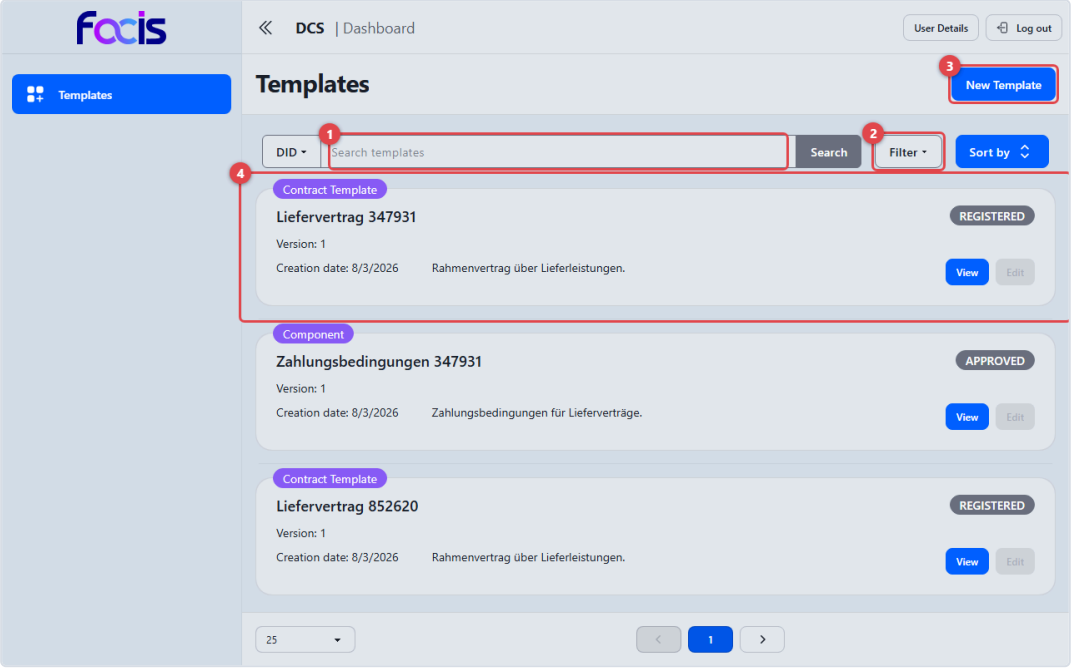
Creating Templates (Template Creator)

As a Template Creator you design the building blocks of contracting: reusable **components** with clauses and machine-readable rules, and **contract templates** that assemble these components into a complete contract document. You submit your drafts for review; only after review, approval and registration can contracts be derived from them.

Prerequisites: You are signed in with the Template Creator role. The side navigation shows the **Templates** area.

The template overview

Open **Templates** in the side navigation.

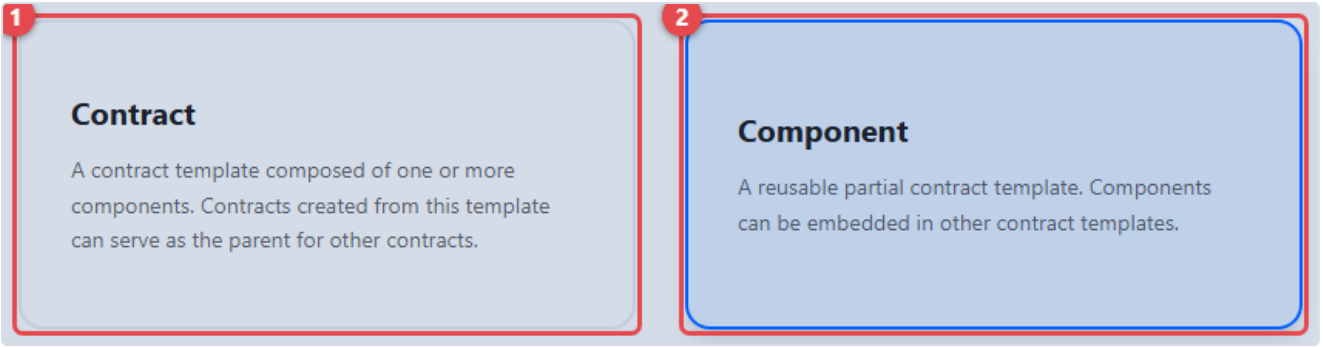


1. Use the search field **(1)** to filter the list. The selector in front of it determines what is searched; the default is the unique identifier (**DID**). The other options are **Name**, **Description**, **Version** and **Template Data**.
2. **Filter (2)** narrows the list by status (for example only DRAFT or SUBMITTED).
3. **New Template (3)** starts a new template.
4. Each entry **(4)** shows name, version, creation date, description and status. **View** opens the detail view; **Edit** is only available for drafts (DRAFT).

Below the list you set the page size and page through the results.

Creating a new template: choosing the type

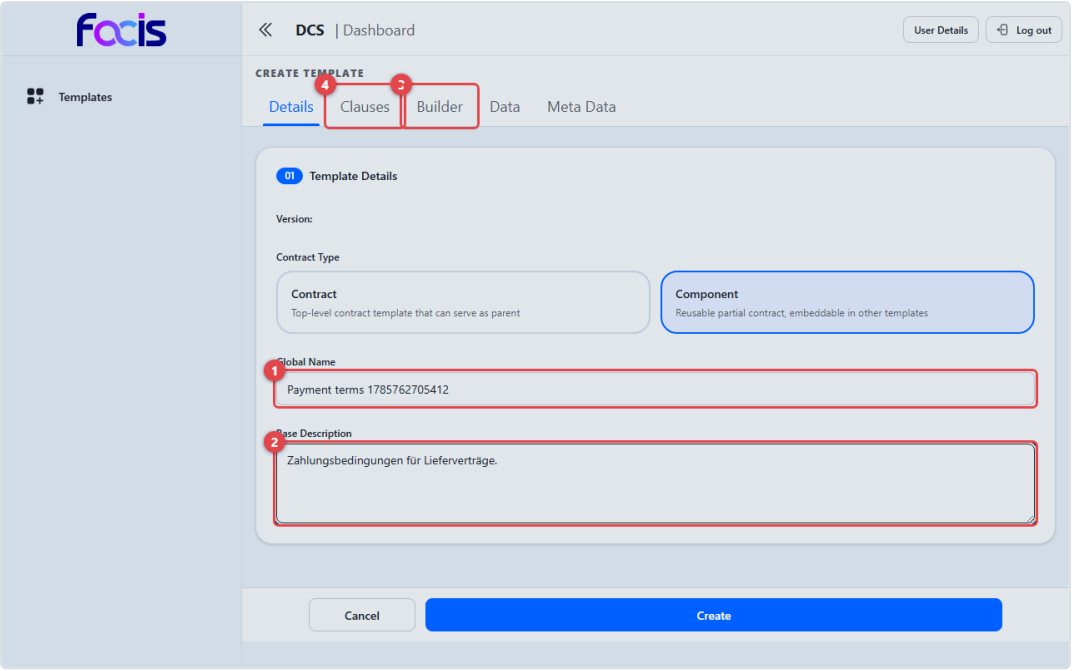
After **New Template** you first choose the template type.



1. **Contract (1)** is a complete contract template from which contracts are later derived. It is assembled from components and can itself serve as the parent template of further contracts.
2. **Component (2)** is a reusable building block (for example payment terms) that is embedded in contract templates.

Entering the basic data (the "Details" tab)

The editing view is organized into the tabs **Details**, **Clauses**, **Builder**, **Data** and **Meta Data**. The action bar with **Cancel** and **Create** stays visible at the bottom edge.



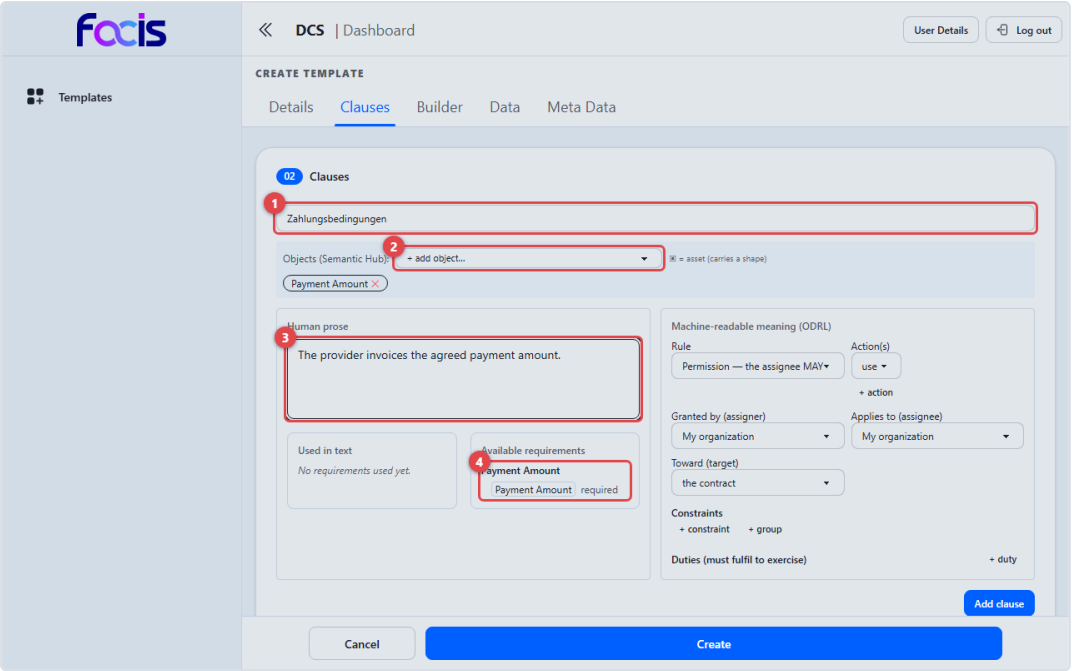
1. Enter a unique name under **Global Name (1)**.
2. Describe the purpose of the template under **Base Description (2)**.
3. Use the **Builder** tab **(3)** to edit the document outline and **Clauses (4)** to edit the clauses (see below). Under **Contract Type** you can check which type you chose.

If the name or the description is empty, **Create** stops: the view jumps to the first incomplete field and reports "Global Name is required." or "Base Description is required." there. Nothing is saved.

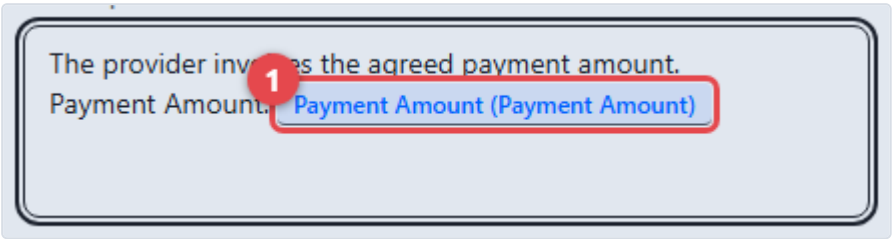
There is no status field. The status of a template changes only through the working steps submit, review, approve and register.

Writing clauses (the "Clauses" tab)

In the clause editor every clause is written in two languages: as **readable text** for people and as a **machine-readable rule** that the system can later check and enforce automatically. Both halves sit side by side.



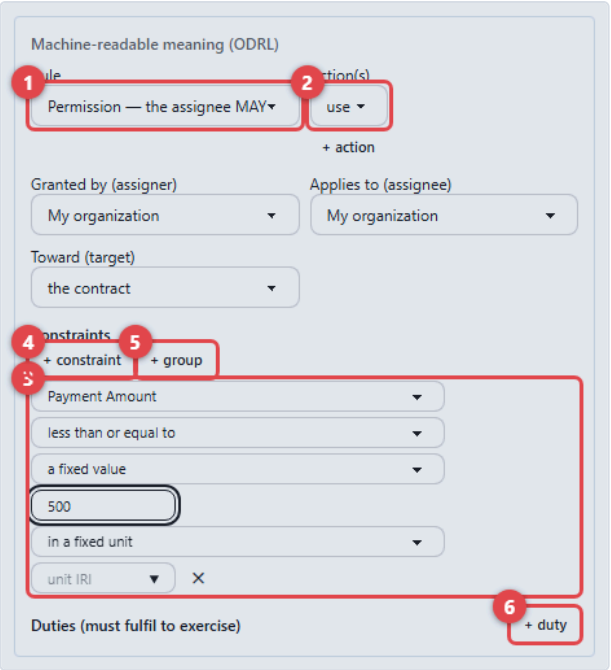
1. Give the clause a title in field **(1)** (for example "Payment terms").
2. Under **Objects (Semantic Hub) (2)** and the **+ add object...** selector you choose the terms the clause refers to, for example **Payment Amount**. The chosen terms appear below as chips that can be removed again with ✕. The terms come from the central vocabulary (Semantic Hub) and are always chosen, never typed freely; this keeps all templates consistent.
3. Write the clause text under **Human prose (3)**.
4. The **Available requirements** field **(4)** lists the chosen terms as insertable building blocks, each with the note **required** for mandatory fields. Clicking an entry inserts a fillable placeholder into the clause text at the cursor.



The inserted placeholder appears in the clause text as its own chip. This is exactly the field the Contract Creator fills in later in the derived contract; for mandatory fields the system enforces the input before the contract can be offered. The **Used in text** field shows which building blocks have already been used.

The machine-readable rule

To the right of the clause text, under **Machine-readable meaning (ODRL)**, you define the formal meaning of the clause.

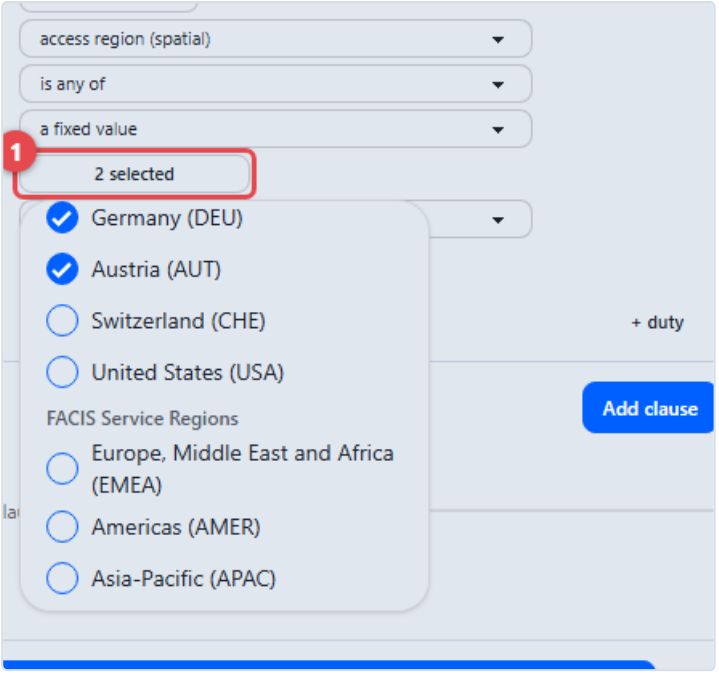


1. **Rule (1)** sets the rule kind. The choices are **Permission** (the addressee may), **Prohibition** (the addressee must not) and **Obligation** (the addressee must).
2. **Action(s) (2)** names the action (for example **use**); **+ action** adds further actions. With **Granted by (assigner)**, **Applies to (assignee)** and **Toward (target)** you define who grants the rule, whom it applies to and what it refers to. The selection offers clear terms for this, such as **My organization**, **The counterparty** and **the contract**.
3. A constraint row **(3)** narrows the rule down. It is read from top to bottom: first the property (for example **Payment Amount**), then the comparison (for example **less than or equal to**), then where the limit comes from. With **a fixed value** you enter the value below yourself; if you choose one of the semantic fields instead, the limit is only fixed during contract negotiation. Below that comes the unit of measure: **in a fixed unit** with a fixed unit in the field next to it, or a semantic field ("in the ...") if the unit is negotiated as well. The unit field suggests the currencies stored in the vocabulary and otherwise stays free; it is optional. The ✕ at the end of the row removes the constraint.
4. **+ constraint (4)** adds another constraint. By default, several constraints must all be satisfied together.
5. **+ group (5)** opens a nested constraint group with its own combination logic. This lets you express things like "purpose is research **and** (region is Germany **or** Austria)".
6. **+ duty (6)** under **Duties (must fulfil to exercise)** attaches a duty to the permission that must be fulfilled to exercise it (for example "must delete afterwards"). A duty can carry its own constraints and, via **+ consequence**, a follow-up duty for the case that it is not fulfilled.

Add clause adds the finished clause to the template; the form clears for the next clause.

Spatial constraints with country lists

With the **access region (spatial)** property you define which countries or regions a rule applies to.



The comparison determines how you select:

- **equal to** allows exactly one country; as long as none is chosen, the selector shows **choose value**.
- **is any of**, **is none of** and **is all of** open a multi-select **(1)**: clicking the field unfolds the list, and entries are ticked. Before the first selection it reads **choose values**, afterwards the number of chosen entries (for example "2 selected").

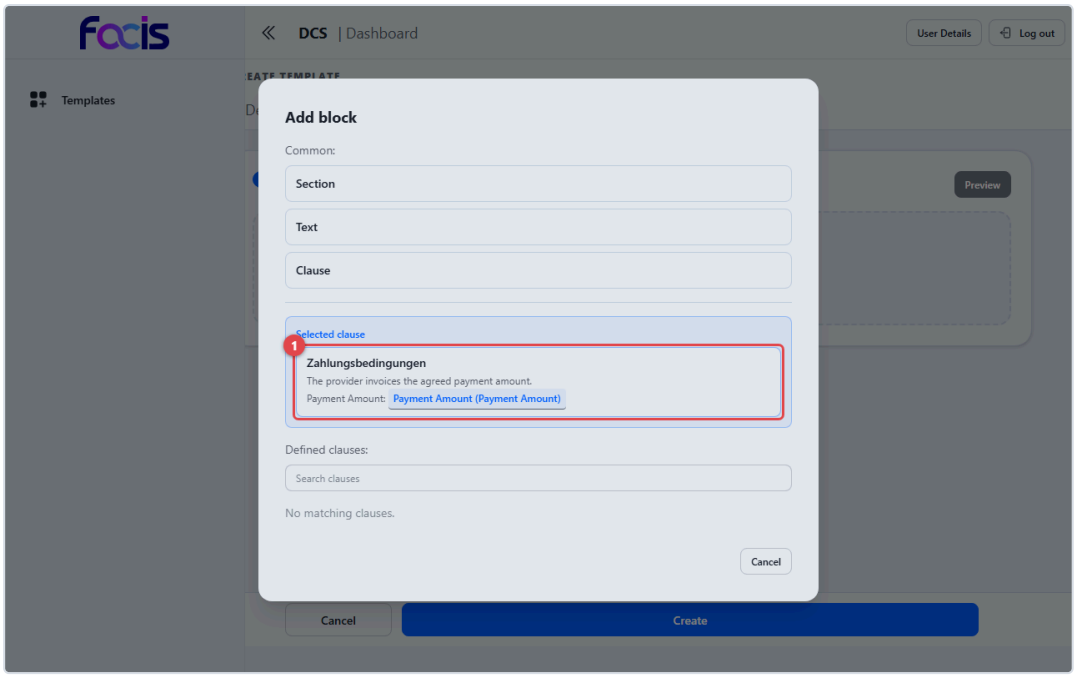
The list comes from the stored vocabulary and is grouped: under **ISO 3166-1 alpha-3 Countries** are countries with their three-letter code (**Germany (DEU)**, **Austria (AUT)**, **Switzerland (CHE)**, **United States (USA)**), under **FACIS Service Regions** are aggregated regions (**Europe, Middle East and Africa (EMEA)**, **Americas (AMER)**, **Asia-Pacific (APAC)**).

Selections you have already made survive a change of comparison. After switching to **equal to**, check which country is selected. If an expected entry is missing, it cannot be added here as free text; the Template Manager maintains the list centrally in the Semantic Hub.

Other vocabulary-backed properties (for example **purpose**) offer their values for selection in the same grouped way.

Placing a clause in the document

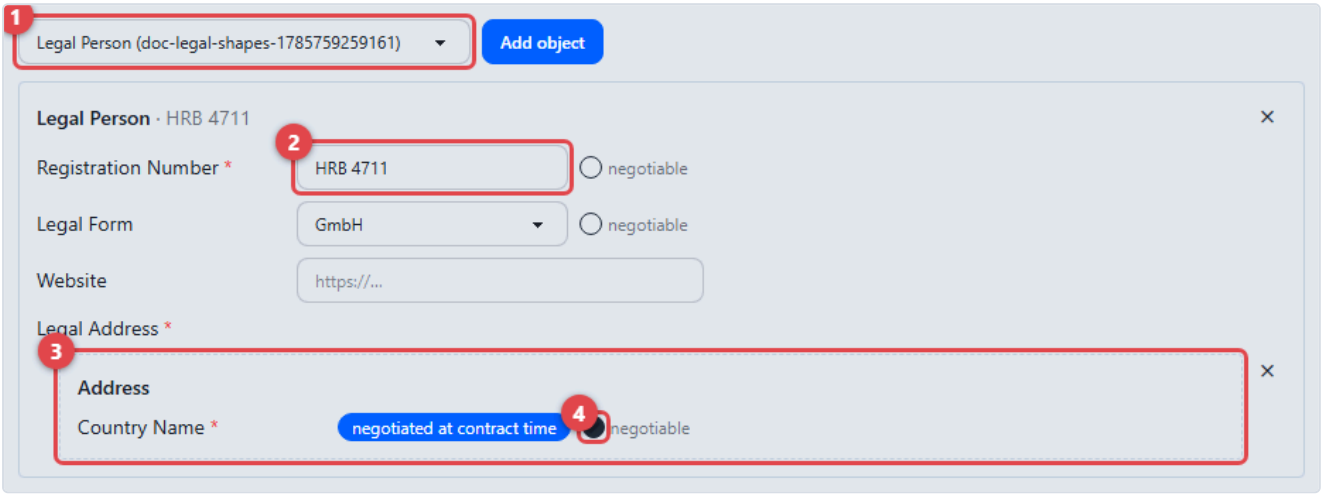
Created clauses only appear in the contract document once you add them to the outline. To do this, choose **Place in document**.



The **Add block** dialog offers the general block types **Section**, **Text** and **Clause** under "Common" and lists the clause you just created under "Selected clause" with its text and its placeholders. Clicking the clause **(1)** adds it as a block to the document outline. Under "Defined clauses" you find existing, not yet placed clauses via search. If the search finds nothing, it reads "No matching clauses."; if there is no unplaced clause at all, the dialog reports "No unplaced clauses from the Clauses tab.". **Cancel** closes the dialog without adding anything.

Semantic data objects (the "Data" tab)

Besides clauses and placeholders, a template can carry structured **data objects**, for example a legal person with its address. Which object kinds are available is determined by the shape libraries registered in the Semantic Hub; the objects are chosen and clicked together, never modelled freely.



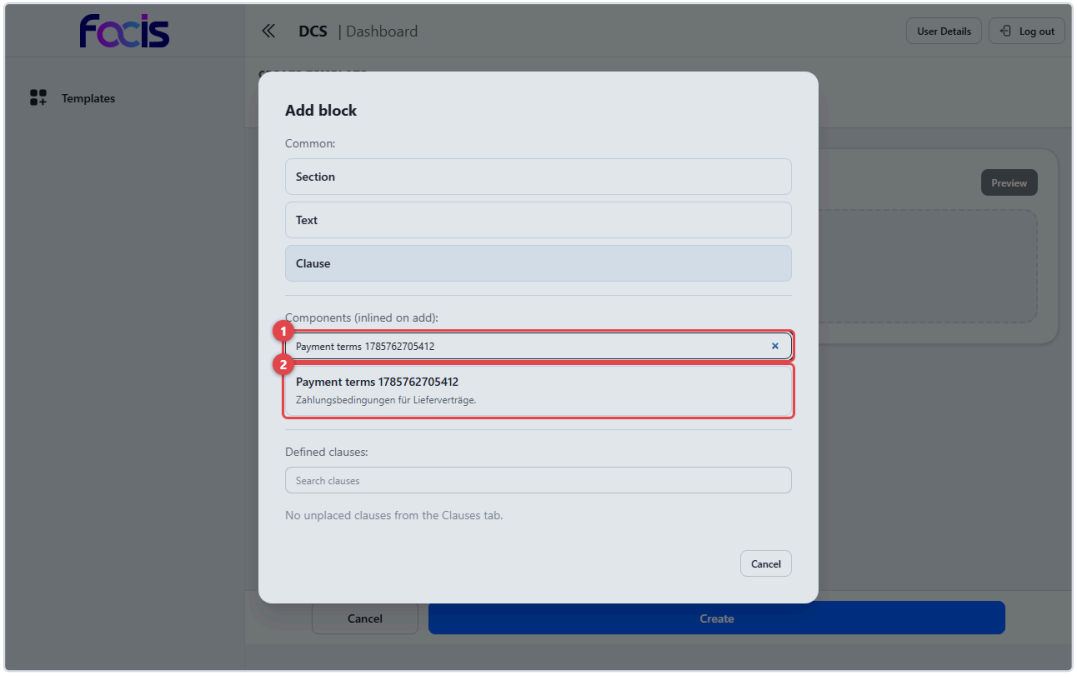
1. In the selector **(1)** choose an object kind from the library (for example a legal person) and add it to the template with **Add object**.

2. The object appears with all fields its shape prescribes. Fixed values (for example a register number) **(2)** are entered directly; they then apply unchangeably to all contracts derived from the template. Fields with a fixed value list appear as a selection, fields for an external address as a reference field.
3. If a field refers to another object (for example the person's address), add it as a nested object **(3)** via the corresponding **Add ...** button and fill it in the same way.
4. With the **negotiable** checkbox **(4)** you mark a field as negotiable. Its value is then not fixed in the template but filled in only in the derived contract; there the field appears as an input field in the contract content and is treated like the other contract fields.

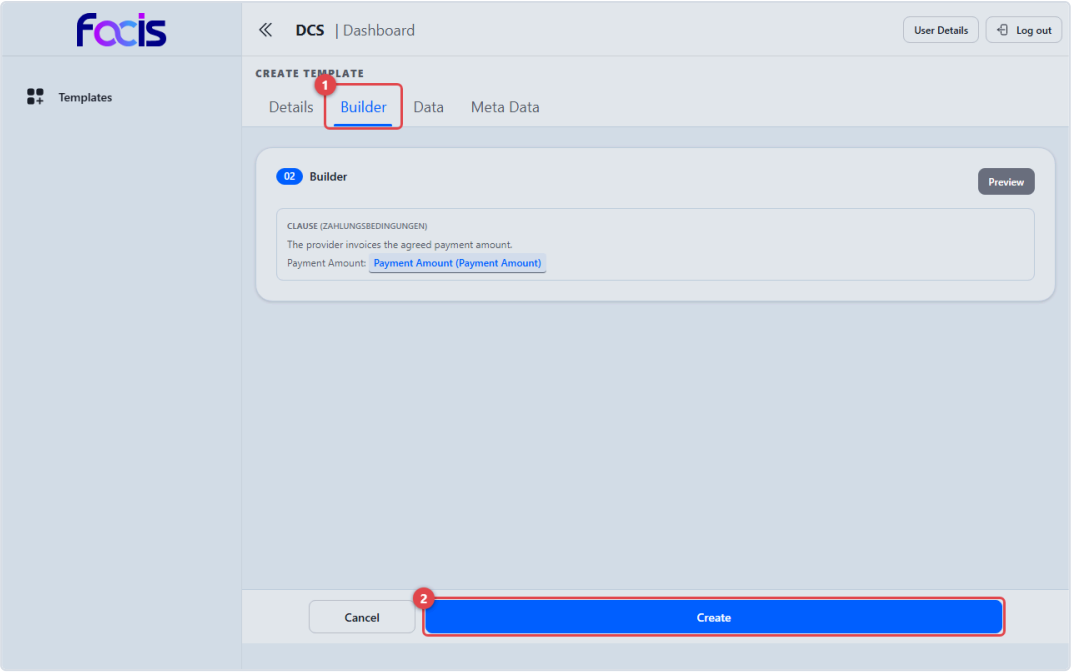
If no shape libraries beyond the document envelope are registered in the Semantic Hub, the tab says so: "No shape libraries are registered in the Semantic Hub beyond the document envelope ...". The Template Manager maintains the libraries (see Managing Templates).

Assembling a contract template from components

Create a new template of type **Contract**, enter the name and description and switch to the **Builder** tab. Use **Add block** to add content.



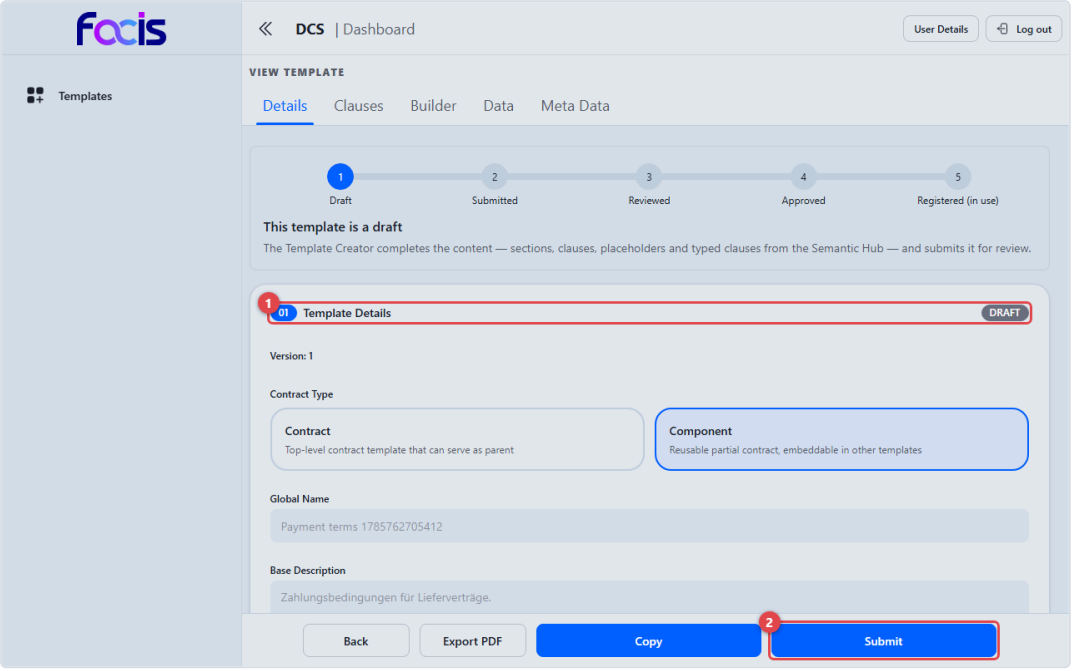
1. In the dialog, under "Components (inlined on add)", use the search field **(1)** to find the approved component you want.
2. Clicking the component **(2)** copies its blocks, placeholders and rules completely into the contract template. Simple block types (for example **Text**) are available next to it for your own sections; a text block is adopted in the dialog with **Confirm**.



The **Builder** tab **(1)** shows the outline of the document with all added blocks, and for each clause its title, text and the placeholders it contains. Use **Preview** to check how the document will look later. **Create (2)** creates the template; **Cancel** discards it. If creation fails, a red error message appears; your input is kept.

Submitting a draft

Open the template via **View** in the overview.



The detail view shows a progress bar across the stations Draft → Submitted → Reviewed → Approved → Registered (in use), below it the note "This template is a draft" with an explanation of the next step. The sections of the template are numbered; **01 Template Details (1)** is the first, with the **DRAFT** status badge to its right.

With **Submit (2)** you submit the draft for review; the template changes to SUBMITTED and appears for the Template Reviewer under **Review Tasks**. From this point on you cannot edit it while the review is running. **Export PDF** produces a PDF version of the current state at any time.

While a template is in the DRAFT status you can keep editing it via **Edit** and save with **Update**; the status stays unchanged. **Copy** creates a copy of the template as a new draft when you need a variant instead of changing the existing one.

When a template is rejected

If the review or the approval rejects your template, it carries the status **REJECTED**. In the detail view, **Submit** is then available again to put the template back into review. If content changes are needed, **Copy** creates an editable copy as a new draft.

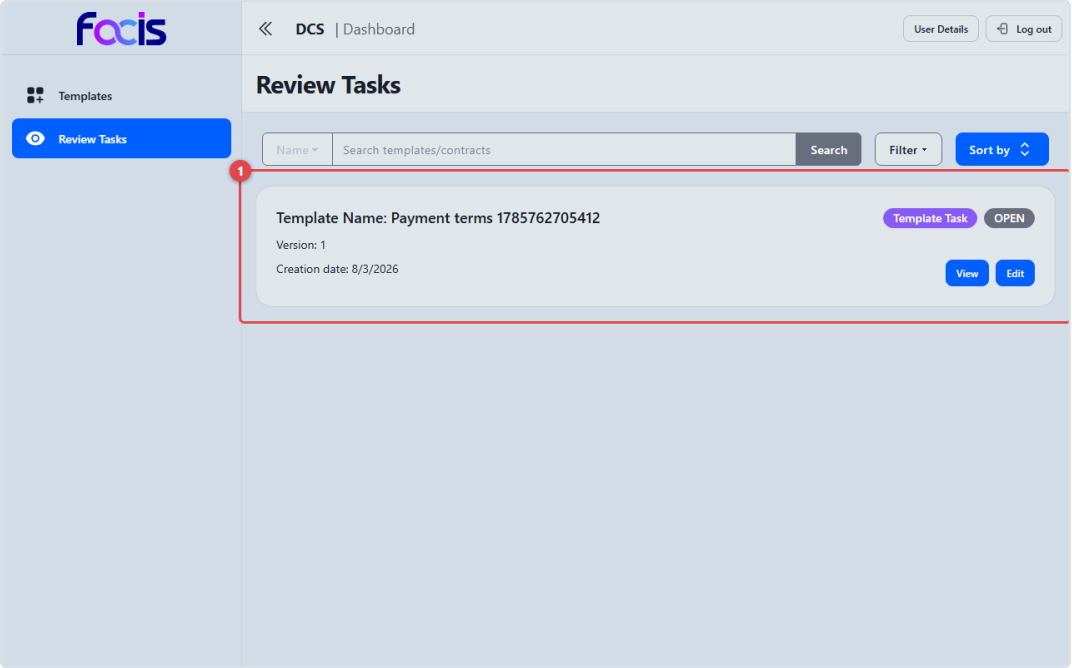
Reviewing Templates (Template Reviewer)

As a Template Reviewer you check submitted templates in substance and form before they move on to approval. The system always runs the formal verification against the stored rule sets itself; it cannot be skipped.

Prerequisites: You are signed in with the Template Reviewer role. The side navigation shows **Templates** and **Review Tasks**.

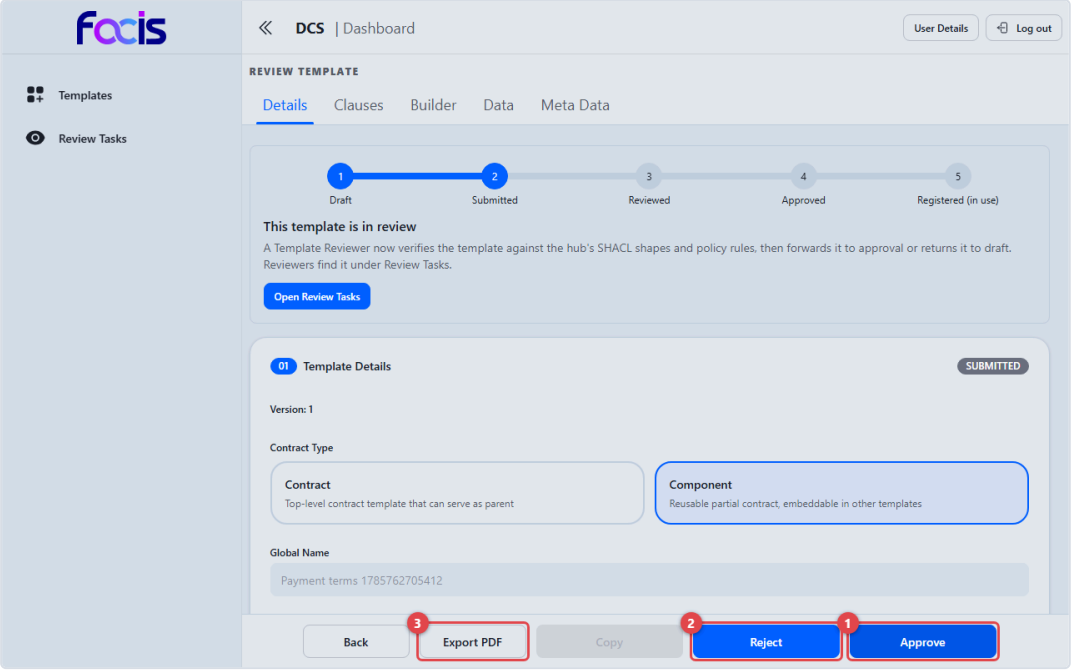
Finding open review tasks

Open **Review Tasks** in the side navigation.



The list **(1)** shows all open review tasks with template name, version and creation date. On the right, a badge marks the kind of task (**Template Task**) and next to it its state (**OPEN**). For a Contract Reviewer the same list also carries the contract tasks. Use the search field, **Filter** and **Sort by** to narrow the list; the search here matches the name, and the selector in front of it has no effect. **View** opens the review view, **Edit** the editing view of the same template.

Reviewing a template



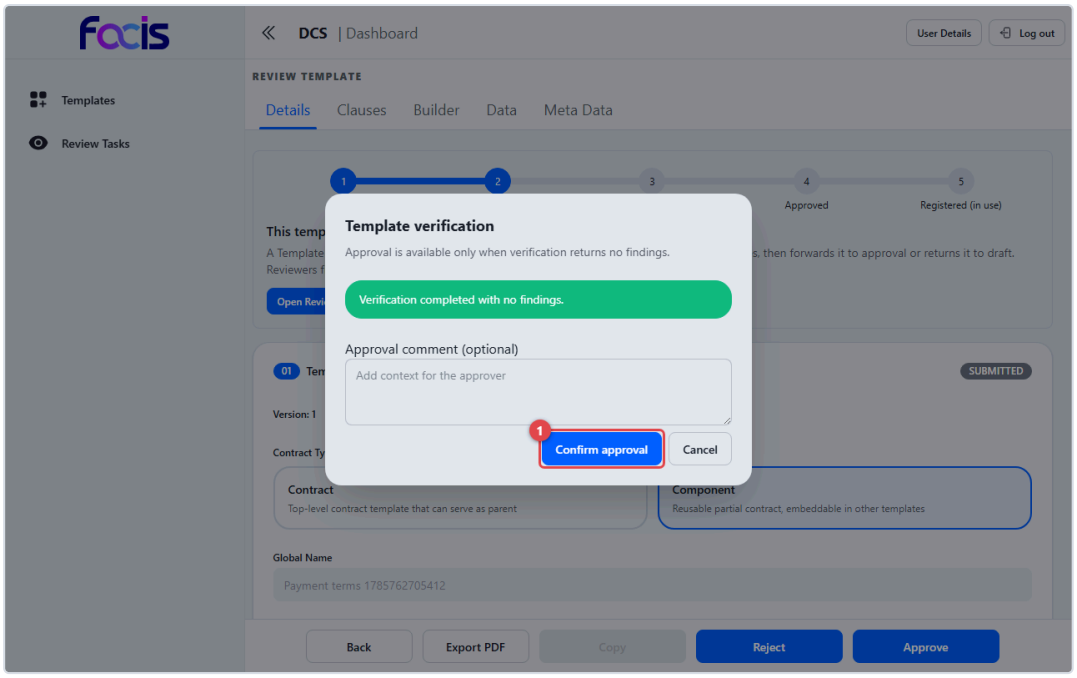
The **Review Template** view shows the template completely and read-only: the tabs **Details**, **Clauses** (for components), **Builder**, **Data** and **Meta Data**, the progress bar with the current status **SUBMITTED** and a note about the next step.

1. **Approve (1)** starts the review and, after successful verification, forwards the template to the Template Approver.
2. **Reject (2)** rejects the template; it goes back to the Template Creator for rework.
3. **Export PDF (3)** produces a PDF version of the state under review. Next to it are **Back** (back to the task list) and **Copy** (a copy of the template as a new draft; locked for a template under review).

There is no separate verify button. Verification is part of **Approve** and runs exactly once on every approval attempt.

The verification result

After **Approve**, the **Template verification** dialog opens. Its subtitle states the condition: "Approval is available only when verification returns no findings."



- **No findings:** A green bar reports "Verification completed with no findings.". Under **Approval comment (optional)** you can leave a comment for the approving person; **Confirm approval** triggers the forwarding.
- **Findings found:** The dialog lists every deviation found (for example "Missing semantic classification") and asks you to resolve them before another approval attempt. No confirm button appears in this case. Close the dialog, have the template reworked and start the approval again.
- **Verification failed technically:** The dialog reports "Verification could not be completed. Retry the verification before approving." and offers **Retry**. The template was not forwarded.

Cancel and the Escape key close the dialog without submitting anything; focus then returns to **Approve**.

Rejecting a template

Reject opens the **Confirmation** dialog with the question "Add comment?". The comment is optional and is stored permanently in the template's audit trail together with the return. **Submit** carries out the return, **Cancel** aborts. The template then carries the status **REJECTED** and can be resubmitted by the Template Creator.

After forwarding, the template changes to the REVIEWED status and appears for the Template Approver under **Approval Tasks**.

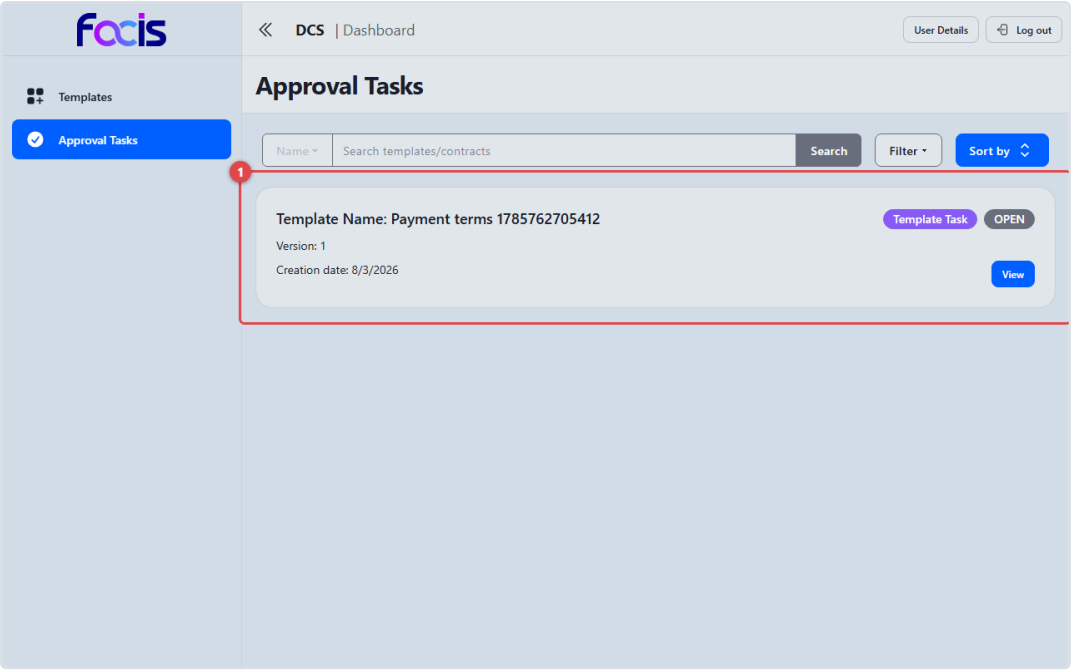
Approving Templates (Template Approver)

As a Template Approver you make the final decision on reviewed templates. Only your approval makes a template usable for registration and thereby for contract creation.

Prerequisites: You are signed in with the Template Approver role. The side navigation shows **Templates** and **Approval Tasks**.

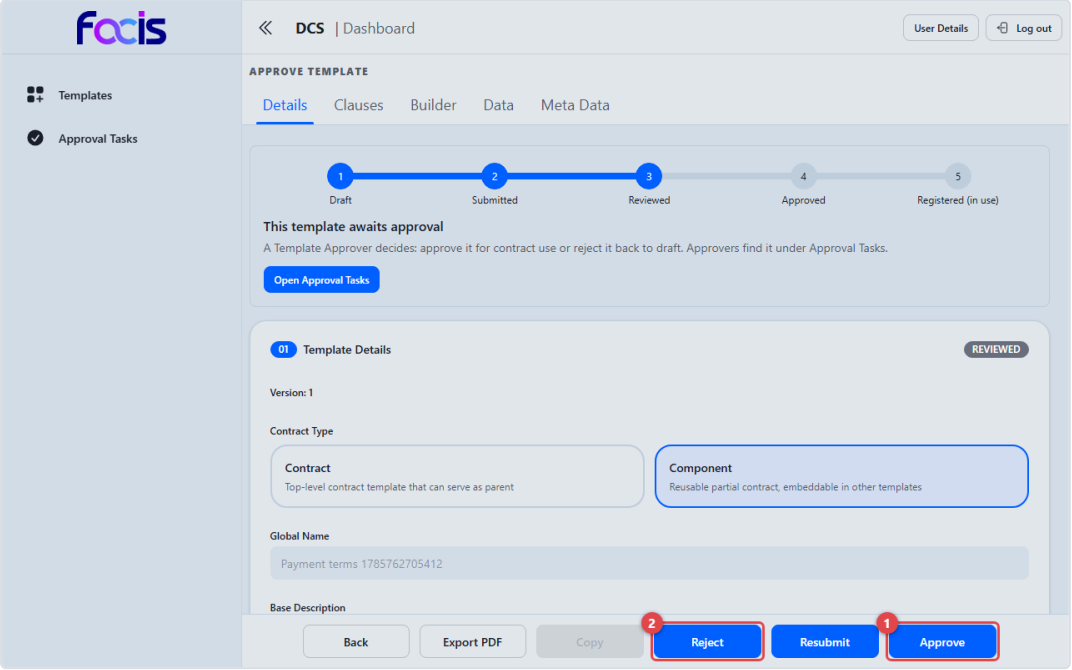
Finding open approvals

Open **Approval Tasks** in the side navigation.



The list shows all templates in the REVIEWED status waiting for your decision, with search field, **Filter** and sorting. **View** opens the approval view.

Approving or rejecting a template

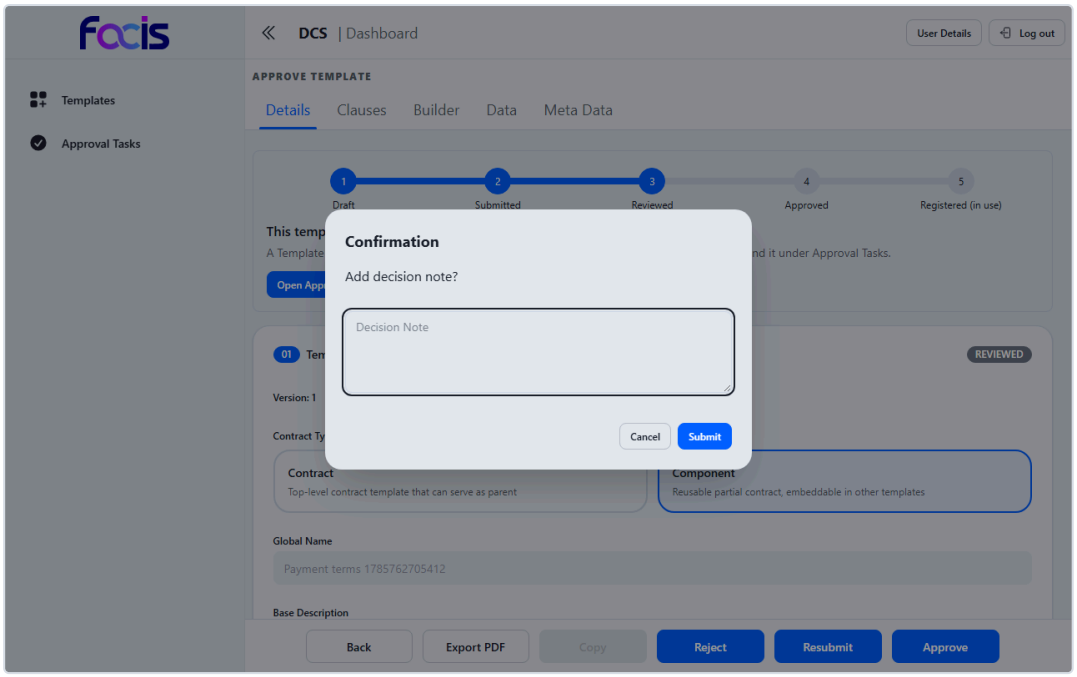


The **Approve Template** view shows the complete, read-only template with the tabs **Details**, **Clauses**, **Builder**, **Data** and **Meta Data**. The progress bar stands at **Reviewed**, the note below it reads "This template awaits approval", and the first section carries the **REVIEWED** status badge on the right.

1. **Approve (1)** gives the template final approval (status APPROVED).
2. **Reject (2)** rejects it; the template goes back for rework.

The same bar holds **Back** (back to the task list), **Export PDF** (a PDF version of the state under approval), **Copy** (a copy of the template as a new draft, locked for you) and **Resubmit**, with which you put the template back into review without rejecting it.

Confirming the decision



Approve and **Resubmit** open the **Confirmation** dialog with the question "Add decision note?". The note under **Decision Note** is optional. **Submit** carries out the decision, **Cancel** aborts.

For **Reject** the same dialog requires a reason under "Add reason:". Without text the rejection does not happen, and the view points out that a reason is required.

After approval the Template Manager can register the template; only then is it available for contract creation (see Managing Templates).

Managing Templates (Template Manager)

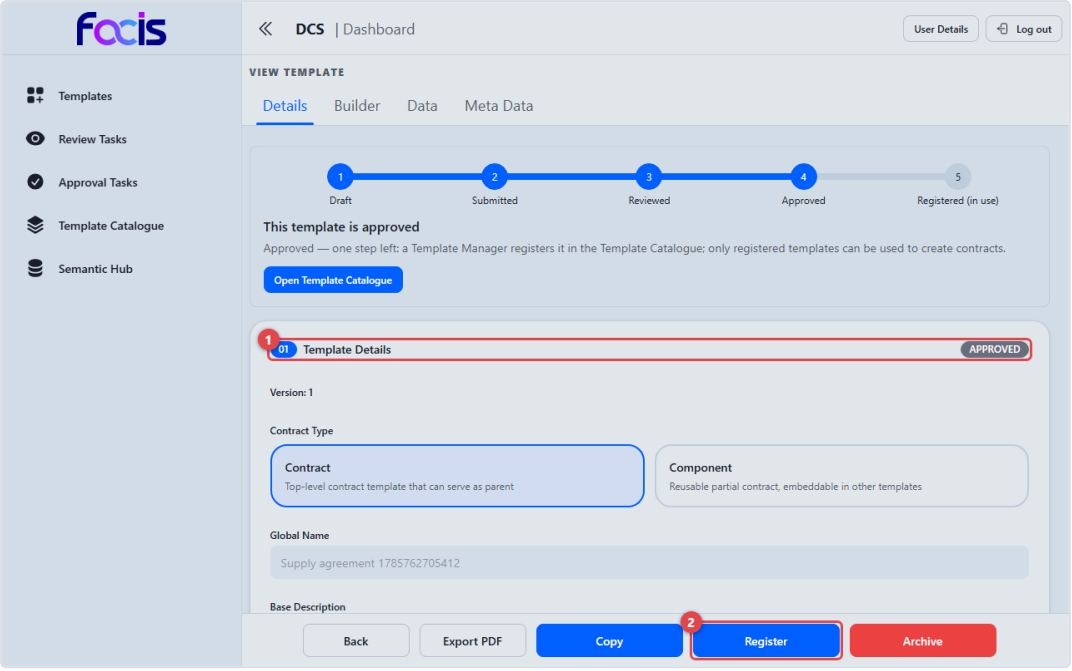
As a Template Manager you make approved templates usable in production: you register them in the template catalogue, keep an eye on the catalogue and maintain, through the Semantic Hub, the central vocabulary that all templates build on.

Prerequisites: You are signed in with the Template Manager role. The side navigation shows **Templates**, **Review Tasks**, **Approval Tasks**, **Template Catalogue** and **Semantic Hub**.

Besides administration, your role also covers the other template steps when needed: you can create templates of your own via **Templates** → **New Template**, review submitted templates and approve reviewed ones. You find the open review and approval tasks under **Review Tasks** and **Approval Tasks**; **View** on a submitted or reviewed template takes you straight to the review or approval view (see Reviewing Templates and Approving Templates).

Registering an approved template

Open the template via **Templates** → **View**.



The progress bar stands at **Approved**. The heading below it reads "This template is approved", and the explanatory text names the remaining step: a Template Manager registers the template in the Template Catalogue, and only registered templates can be used for contracts. The first section of the template **(1)** carries the **APPROVED** status badge on the right.

With **Register (2)** you publish the template in the template catalogue; a confirmation dialog asks before execution ("Proceed with registration?"), which you complete with **Confirm**. Next to it are **Copy** (copy the template as a new draft), **Archive** (take the template out of the active stock), **Export PDF** and **Back**. The recorded history of a template (the **Audit History** tab) is reserved for the Auditor and Compliance Officer roles.

After registration the template carries the status REGISTERED (in use) and can be chosen by Contract Creators as the basis of new contracts. Only from this point on does it appear in their template selection.

Register is only available for templates of type Contract. A component is not registered; it comes into use through the contract template that embeds it.

On a registered template the **Publish** button appears. It puts the template into the federated catalogue so other organizations can find it; here too you confirm with **Confirm**. The template then carries the status PUBLISHED and remains usable for contracts.

Retiring a template (Archive)

Archive takes a template out of the active stock; the confirmation dialog asks "Proceed with archiving?" and is completed with **Confirm**. The effect depends on the state of the template:

- A **registered** (or already published) template receives the status **DEPRECATED**. No new contracts can be derived from it any more; it disappears from the Contract Creators' template selection. Existing contracts continue unchanged and carry the badge "This contract uses a deprecated template" in the contract overview.
- A template that is not yet registered receives the status **DELETED** and is thereby removed from the active stock.

For an already retired template (DELETED or DEPRECATED), **Archive** is no longer offered; retiring it a second time is not possible.

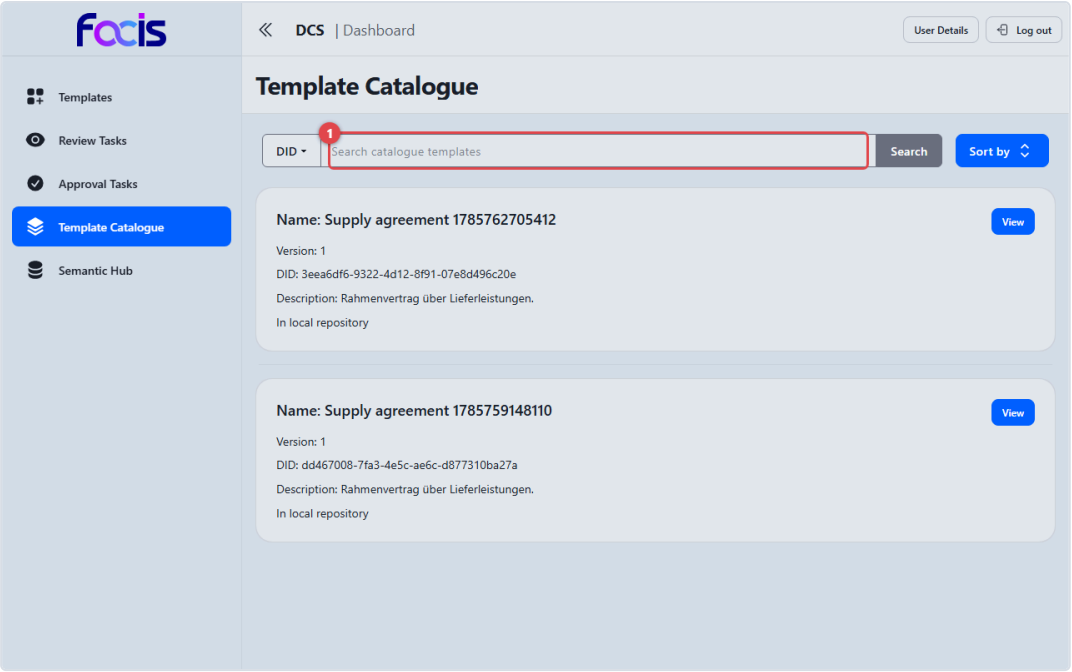
Updating a registered template

A registered template is not changed directly. Use **Copy** to create a copy as a new draft; it passes through review and approval and is then registered as the next version of the same template. Older versions remain available through the history, and contracts already created stay valid unchanged.

Once the new version is registered, the predecessor carries the badge "A newer template version is available" in the template overview as a link to the new version. The same badge appears in the contract overview on contracts still based on the predecessor.

Template catalogue

The **Template Catalogue** area lists the templates published in the federated catalogue.



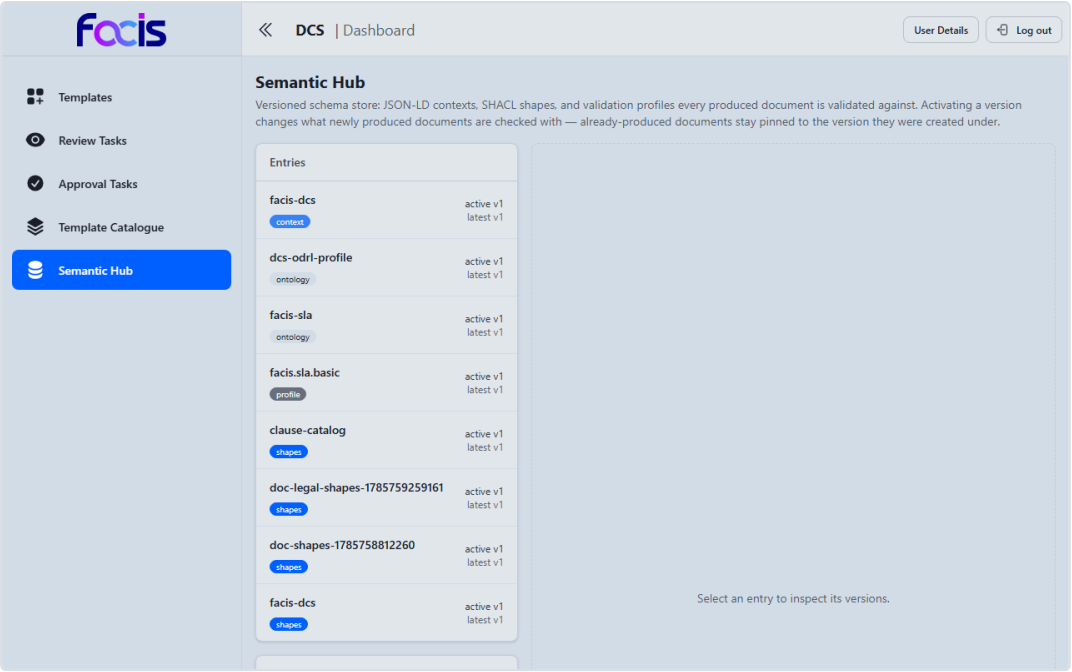
Use the search field **(1)** to find a template; the selector in front of it switches between searching by **DID** and by **Name**, and **Sort by** changes the sorting. Each entry **(2)** names name, version, identifier and description. The last line of an entry says where the template comes from: **In catalogue** means it lives in the federated catalogue but not in your own stock; **In local repository** means you already have it.

View (3) opens the detail view with the tabs **Details**, **Meta Data** and **Preview**. Under **Preview** you read the document of the catalogue version before adopting it. **Register** adopts a foreign template into your own stock; after the confirmation with **Confirm**, the DCS creates a local template from it that receives its own identifier.

If no federated catalogue is set up in your installation, the area stays empty, and publishing and catalogue search are not available. Whether and how a catalogue is operated is decided by the operator of your installation. If the catalogue rejects the retrieval, an error message appears instead of the list; contact the operations team in that case.

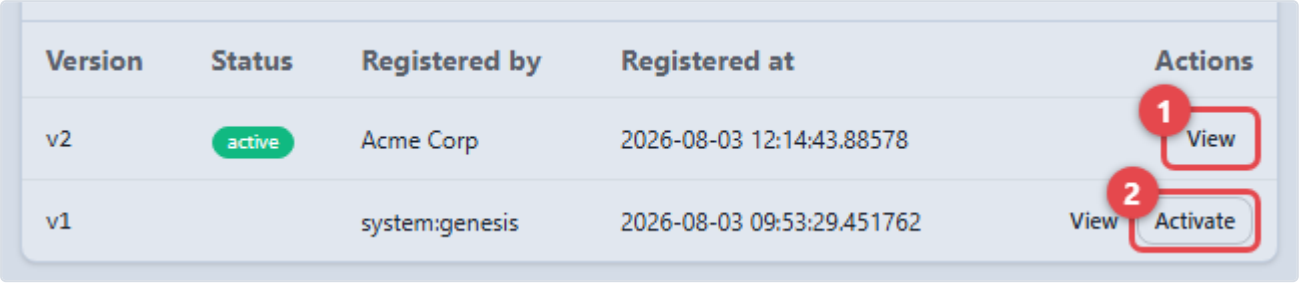
Semantic Hub: maintaining the vocabulary

The Semantic Hub is the central registry of terms, rule sets and validation schemas that the clause editor draws its selection lists from: from the country lists of spatial constraints to the shape libraries for semantic data objects. New entries can be published during operation, without a software change.



On the left, under **Entries**, the view lists the registered entries with their kind (**context**, **ontology**, **shapes**, **profile**) and their version state, for example "active v1" and "latest v1"; without an activated version it reads "no active version". As long as no entry is chosen, the right side reads "Select an entry to inspect its versions."

Clicking an entry shows its versions on the right as a table with **Version**, **Status** (the **active** badge on the effective version), **Registered by** and **Registered at**.



Two actions are available per version:

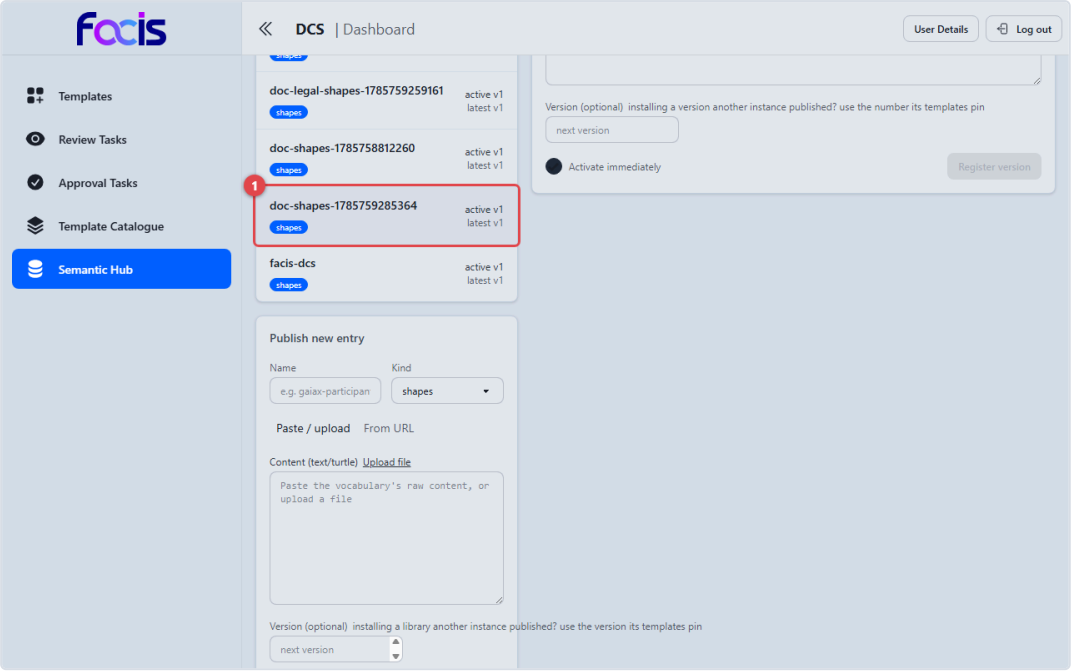
- **View (1)** shows the stored content of the version.
- **Activate (2)** makes another version the effective one. This is also how you return to an earlier version: new documents are then created against that state again, while documents already created stay bound to the version under which they were created. The already effective version lacks the button; it cannot be activated again.

Below the table, **Register new version** registers another version of the same entry; the form matches the publish form below, with the **Activate immediately** checkbox.

Which rule version is checked for a contract does not follow the currently active version: every contract is bound to the rule state that was active when it was created, and it is checked against exactly that state later as well. A stricter new version therefore only affects documents created after it.

Below the list, under **Publish new entry**, you publish new entries:

1. Enter a unique name under **Name**.
2. Under **Kind** choose the kind of entry (for example **shapes** for a shape library from which Template Creators click together data objects).
3. Bring in the content. The **Paste / upload** tab accepts it directly in the **Content** field; with **Upload file** next to it you upload a file instead. The **From URL** tab fetches the content from an address under **Source URL** and stores it as a version.
4. Under **Version (optional)** enter a fixed version number when you install a library published elsewhere in exactly the version its templates refer to. Without an entry, the next version is created.
5. **Activate immediately** determines whether the new version becomes active right away.
6. **Publish entry** publishes the entry.



The new entry **(1)** then appears in the list with the **active** note and is available system-wide from that moment. In the example it is a shape library from which Template Creators click together data objects in the **Data** tab. An already published version is kept; publishing again creates a further version. Documents already created stay bound to the version under which they were created.

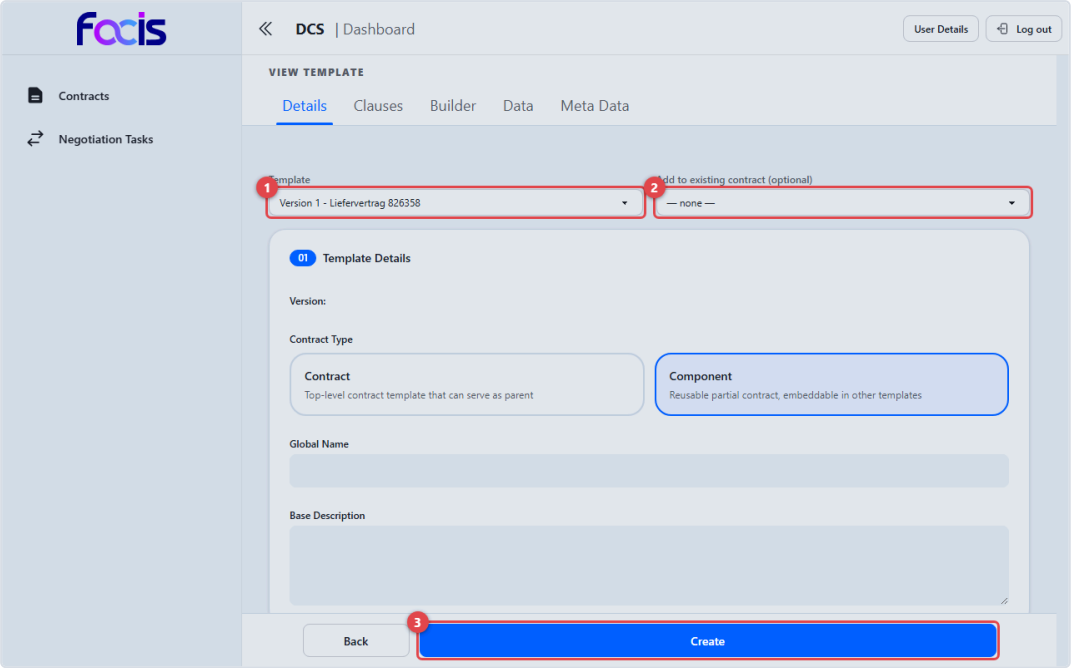
Creating and Negotiating Contracts (Contract Creator)

As a Contract Creator you derive concrete contracts from registered templates, fill in the negotiable values, offer the contract to the counterparty and accompany the negotiation until agreement.

Prerequisites: You are signed in with the Contract Creator role. The side navigation shows **Contracts** and **Negotiation Tasks**. At least one registered contract template exists.

Creating a new contract

Open **Contracts** and choose **New Contract**.

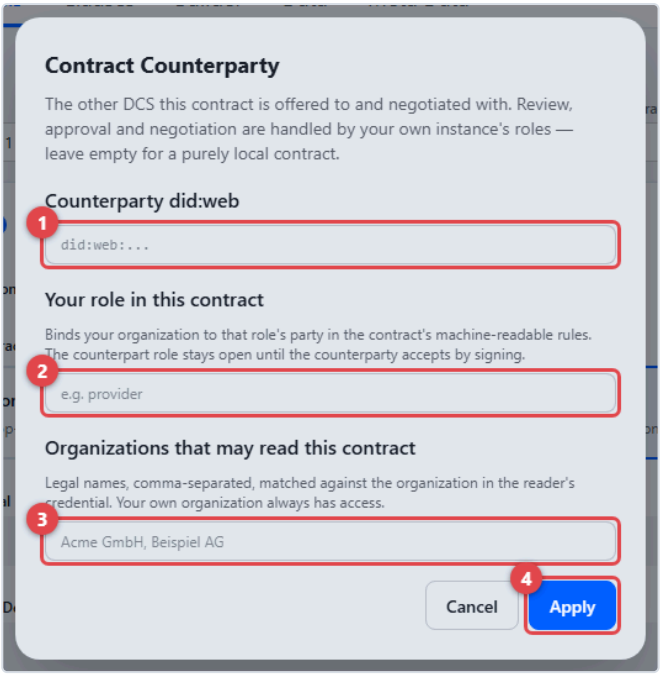


1. Under **Template (1)** choose the registered template, listed with its version ("Version 1 – ...").
2. Via **Add to existing contract (optional) (2)** you assign the new contract to an existing contract as a sub-contract if needed; the top entry of the list ("none") stands for a standalone contract. The assignment later appears in the contract's detail view in the **Part of Contract** card and in the **Structure** tab (see Managing Contracts).
3. **Create (3)** starts the creation, **Back** returns to the contract overview.

Below the two selectors sits a preview area with the tabs **Details**, **Clauses**, **Builder**, **Data** and **Meta Data**. It stays empty because reading a template is reserved for the template roles; this has no consequences for the contract. You read the full contract text right after creation in the **Contract Content** tab.

If the template you want does not appear in the selection, it is not yet registered; approval alone is not enough (see Managing Templates). If there is no usable template at all, the selection reads "No templates available".

Setting the counterparty

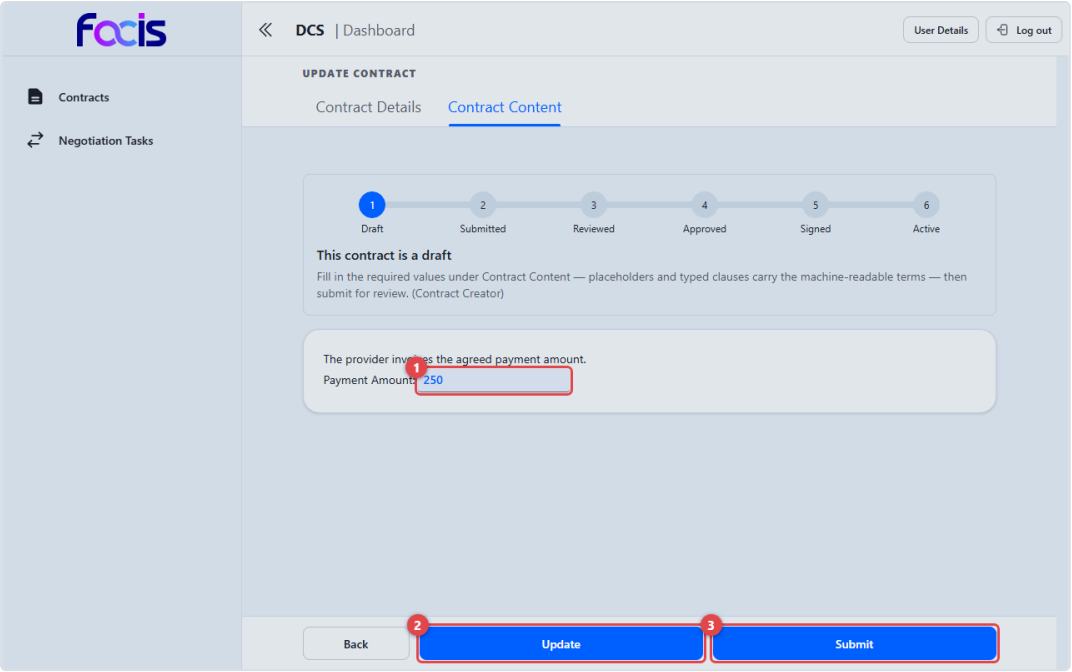


Before creation, the **Contract Counterparty** dialog asks about the parties:

1. Under **Counterparty did:web** enter the unique identifier of the partner organization if the contract is to be negotiated with another DCS. Leave the field empty to create a purely internal contract without a counterparty; the dialog points this out.
2. Under **Your role in this contract** you set which contract role your organization takes (for example provider or customer). If the template names its contract roles, choose from those; otherwise enter the role as text or leave it open (**Not declared**). The entry anchors your organization in the machine-readable contract rules; the counterparty's role stays open until they accept the contract by signing it.
3. Under **Organizations that may read this contract** you name, if needed, further organizations allowed to read the contract, by their official name, separated by commas. The match is made against the organization in the reading person's credential; your own organization always has access. Anyone not listed here and not involved is rejected on access.
4. **Apply (4)** creates the contract; it starts in the **DRAFT** status. Review, approval and negotiation are always handled by the roles of your own organization. **Cancel** aborts the contract creation.

Filling in contract content

Open the contract for editing (**Contracts** → **Edit**).



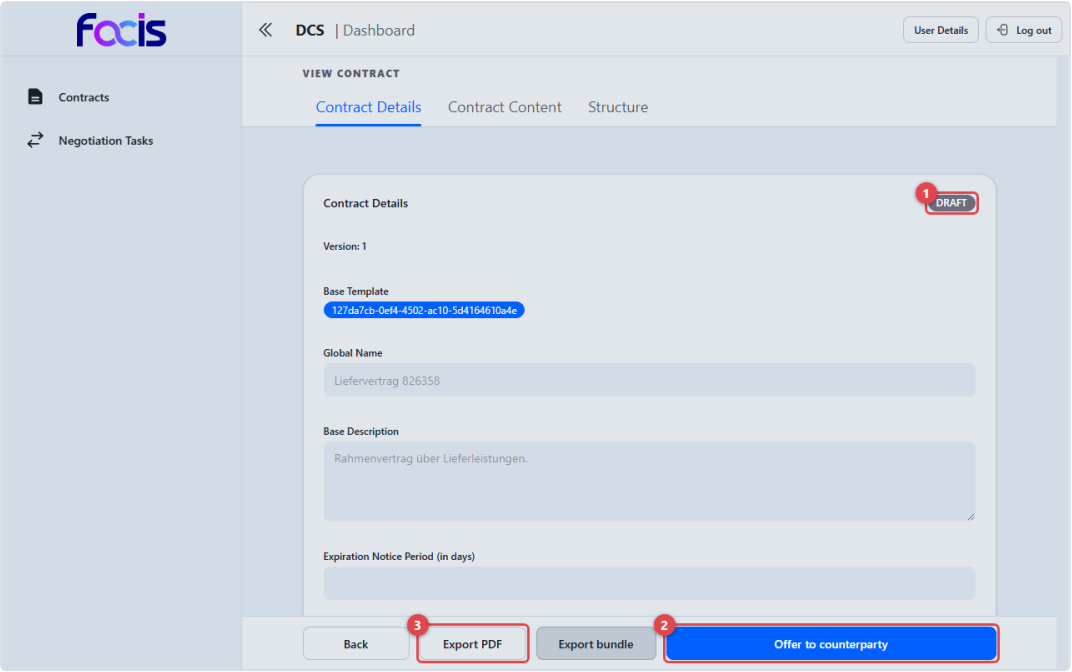
The editing view leads through the tabs **Contract Details** and **Contract Content**; the progress bar shows the six stations Draft → Submitted → Reviewed → Approved → Signed → Active, below it a note about the current state ("This contract is a draft") and the next step.

1. Under **Contract Content** you see the contract document with the clause texts. If the template contains fillable placeholders (for example a payment amount), they appear here as input fields **(1)**. Fill in all mandatory fields.
2. **Update (2)** saves the draft without changing the status.
3. **Submit (3)** saves and submits the contract into internal review; this starts the negotiation phase.

If the template carries structured data objects (for example a legal person with an address, see Creating Templates), they also appear under **Contract Content**: fixed values are visible but unchangeable, and fields marked as negotiable are filled in here like the other input fields.

Offering the contract (transmission to the counterparty)

Open the contract view (**Contracts** → **View**).



1. The status badge **(1)** shows the current state (here **DRAFT**).

2. **Offer to counterparty (2)** transmits the draft to the registered counterparty for the first time (status DRAFT → OFFERED). The counterparty receives the complete contract document as a PDF with the embedded provenance and change history.

   Important: As long as a mandatory field of the contract is not filled in, the button stays greyed out. A tooltip names the missing fields ("Fill the required field(s) before offering to the counterparty: ..."). An offer must be complete before it leaves the house. After offering, the button disappears; an offer is made exactly once.

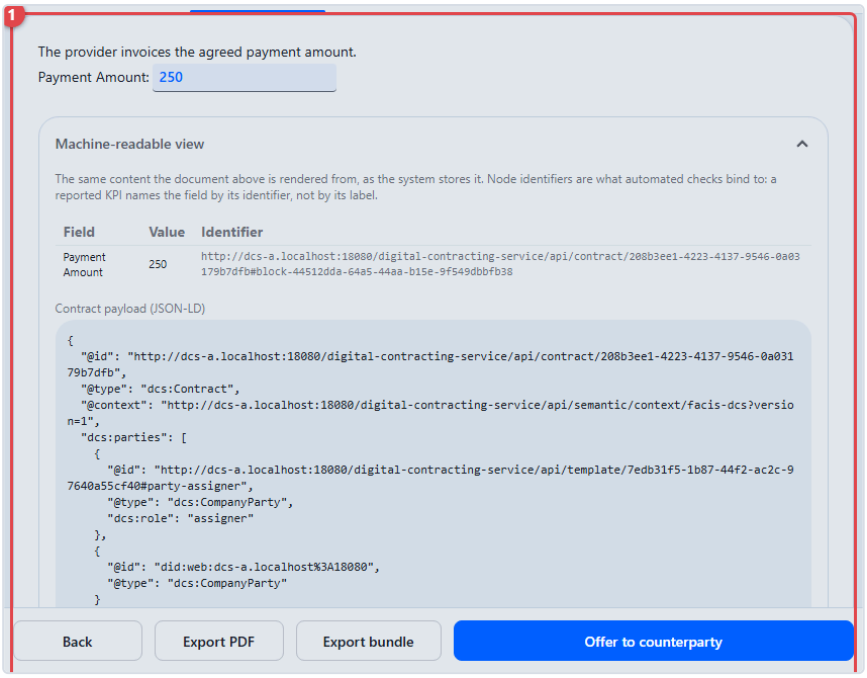
3. **Export PDF (3)** downloads the current contract document as a PDF; **Export bundle** produces a complete package of the contract.

Once the contract is offered or under negotiation, two buttons join the same bar:

- On an offered contract the first is called **Review offer**, during negotiation **Negotiate**; both lead into the negotiation view.
- **Withdraw** withdraws the contract. As a Contract Creator the action is available to you in the states OFFERED, NEGOTIATION, SUBMITTED and REVIEWED; after approval (APPROVED) it is not. The confirmation dialog asks "Withdraw this contract from the counterparty? It cannot be taken forward afterwards." After the confirmation the contract carries the status **WITHDRAWN** and cannot be taken forward; the view returns to the contract overview.

The machine-readable side of the contract

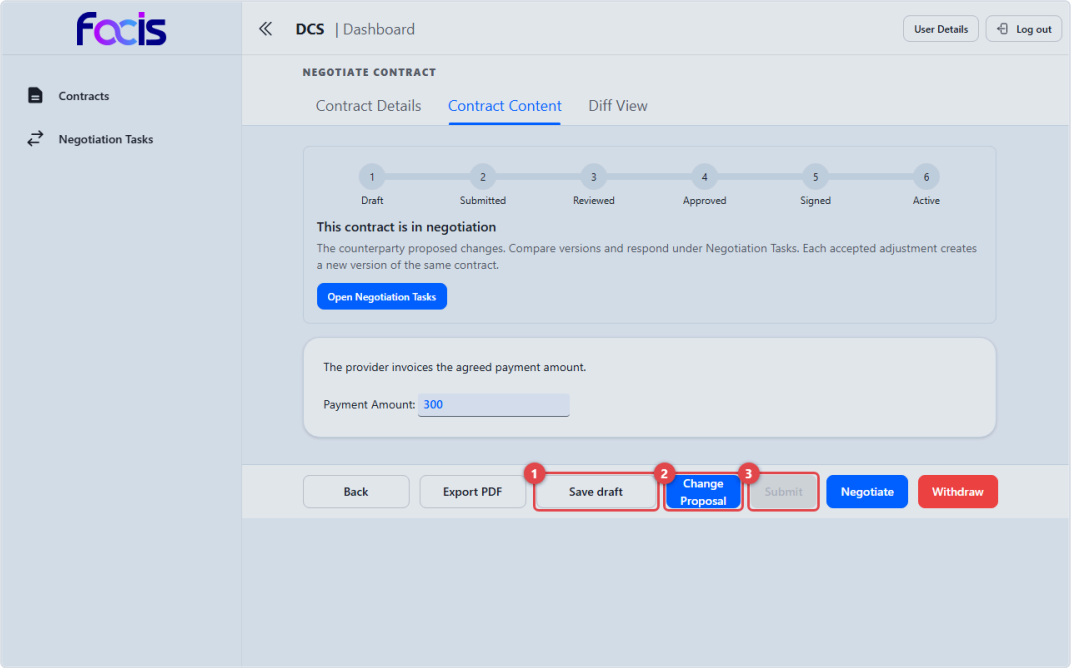
In the **Contract Content** tab, below the readable document, sits the collapsed **Machine-readable view** area. Clicking the heading unfolds it.



The unfolded area **(1)** shows under **Contract payload (JSON-LD)** the complete contract content as the system stores it. If the contract carries fillable fields, a table sits above it with the label (**Field**), the value (**Value**) and the technical identifier (**Identifier**) per field. Document and machine-readable view come from the same contract content and therefore cannot differ. You do not need the identifiers day to day; automatic checks and later reported KPIs refer to them.

Negotiating

After submitting, or after a change by the counterparty, negotiation takes place in the **Negotiate Contract** view, reachable via **Negotiation Tasks** or via **Negotiate** or **Review offer** in the contract view.



The view shows the progress bar, a note about the current state ("This contract is in negotiation") and the tabs **Contract Details**, **Contract Content** and **Diff View**. Under **Diff View** you compare contract versions; under **Contract Content** you change negotiable values. **Save draft** and **Change Proposal** stay greyed out until you have actually changed a value.

1. **Save draft (1)** saves changed values as a **private negotiation draft**: the counterparty sees none of it, and the draft is kept even if you leave the view. On the next open the values are restored. Private here means private to your organization. Colleagues with access to the negotiation see the same draft and can continue or discard it.
2. **Change Proposal (2)** makes the change binding: the contract immediately receives a new version with the changed content, and the replaced version stays in the history. For a contract with a partner organization, the counterparty receives the changed contract document. The proposal itself additionally appears under **Active negotiations** and waits there for the decision (see below). A saved draft is consumed by this.
3. **Submit (3)** closes the negotiation round and hands the agreed state into internal review. The button is locked while a changed input has not been transmitted yet, while unanswered change proposals are open and while a comparison is open. Only when none of this applies does the round count as agreed.

While a private draft is saved, the **Discard draft** button joins the same action bar between **Save draft** and **Change Proposal** and discards the draft on request. It disappears again once the draft has been discarded or transmitted as a change proposal.

On the left of the same bar sit **Back** and **Export PDF**, on the right **Negotiate** (opens the negotiation view again) and **Withdraw** (withdraws the contract, see above).

Accepting a received offer (Accept offer)

If a partner organization offers your organization a contract, it appears in your contract overview with the status **OFFERED**. It is not listed under **Negotiation Tasks** at that point: a negotiation task only comes into being when your side enters the round.

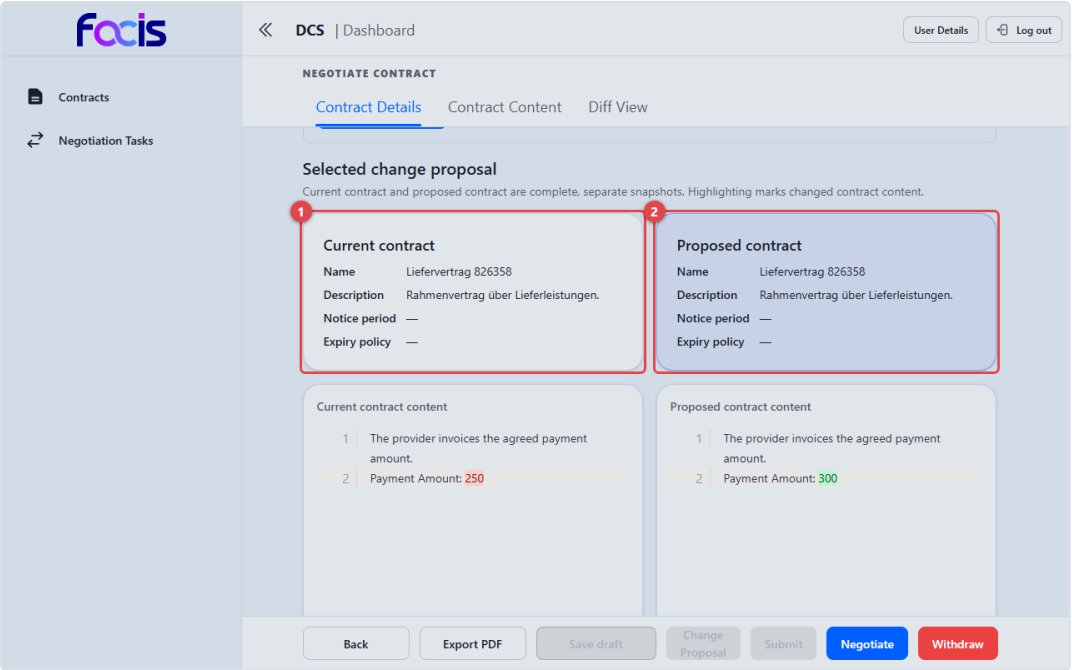
1. Open the contract via **Contracts** → **View** and choose **Review offer** there. The negotiation view opens.
2. To accept the offer unchanged, choose **Accept offer**. The contract changes to the **NEGOTIATION** status, and the round appears under **Negotiation Tasks** from now on. You then close the round with **Submit**; the contract goes into your organization's internal review and approval.
3. To change something instead, adjust the values under **Contract Content** and transmit the change with **Change Proposal**. This also puts the contract into negotiation.

Your own offer cannot be accepted this way: acceptance is reserved for the counterparty; on the offering side it is rejected.

Responding to change proposals

Open proposals appear in the negotiation view under **Active negotiations** as a list, each with the decisions of the parties. **Show** opens the comparison under **Selected change proposal**: the **Current contract** card on the left, **Proposed contract** on the right, below them the differences in contract content.

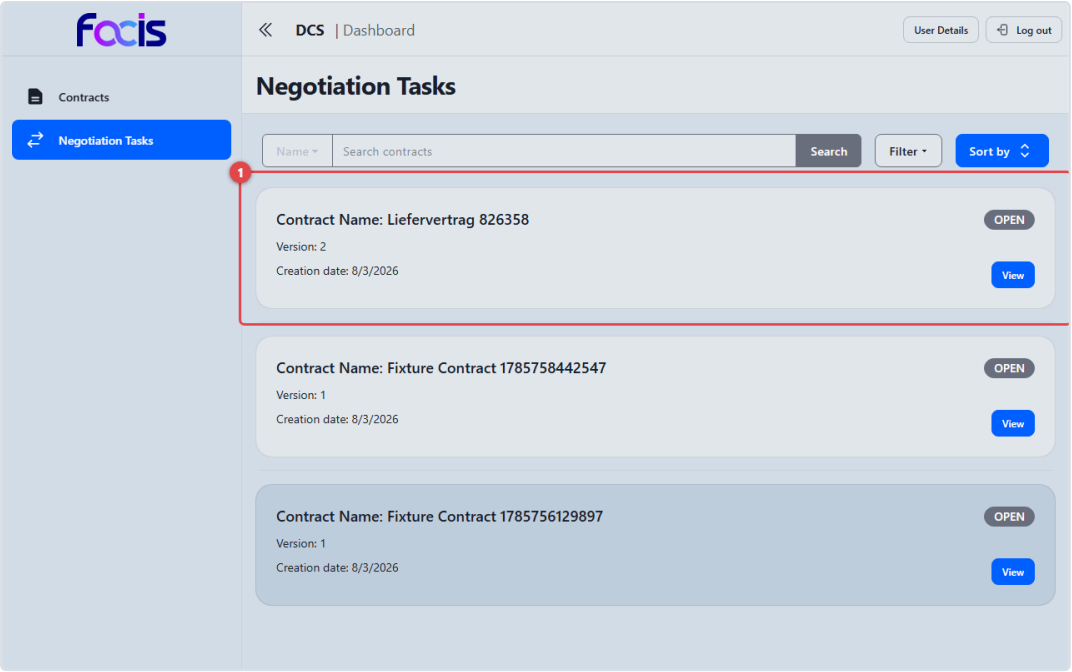
For a proposal already adopted into the contract, **Current contract** shows the version the proposal started from. **Hide** closes the comparison again.



The **Current contract** card **(1)** and the **Proposed contract** card **(2)** sit side by side, each with name, description, notice period and expiration policy. Below them the contract contents face each other: **Current contract content** on the left, **Proposed contract content** on the right, numbered line by line. The changed value is highlighted on both sides, in the example the payment amount 250 versus 300.

In the opened comparison, two decisions are available. **Accept** asks "Accept this change request?" and is carried out with **Confirm**. **Reject** requires a rejection reason in the dialog and is carried out with **Submit**. You cannot accept your own proposal; another authorized person decides it (Contract Creator, Contract Negotiator, Contract Reviewer or Contract Manager).

If a proposal accepted later sets the same entries again, the earlier proposal carries the **SUPERSEDED** badge in the list with a note about which of its entries the agreed contract no longer carries and which proposal overwrote them.



The **Negotiation Tasks** area lists the negotiation rounds as a task list, per entry with name, version, creation date and the task state. Without a filter the list shows the open tasks (**OPEN**), that is the rounds still waiting for an answer from your side; via **Filter** you can also show completed ones (**ACCEPTED**). **View** opens the negotiation view; the search field and **Sort by** narrow the list further.

After negotiation

With **Submit** the contract moves to review (Contract Reviewer) and then to approval (Contract Approver), see the chapters Reviewing Contracts and Approving Contracts. If each side negotiates with its own DCS, each organization runs its own review and approval; one side's approval does not replace the other's.

Closing the round with **Submit** is at the same time your organization's binding confirmation that it agrees to exactly this contract state. It is transmitted to the counterparty, and only when both sides have confirmed the same state does the system allow signing (see Signing Contracts). If your Contract Reviewer rejects the submission, that takes the confirmation back; after renewed negotiation both sides confirm the new state.

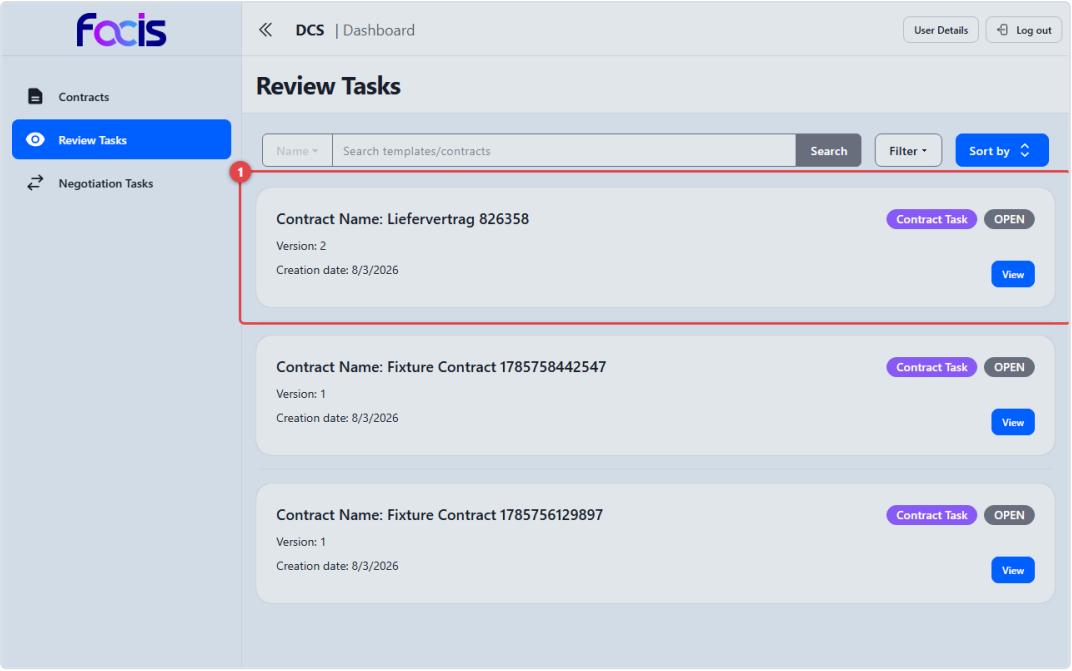
Reviewing Contracts (Contract Reviewer)

As a Contract Reviewer you review negotiated contracts before they move on to approval. You see the complete, agreed contract state and decide whether it goes into approval or returns for rework.

Prerequisites: You are signed in with the Contract Reviewer role. The side navigation shows **Contracts**, **Review Tasks** and **Negotiation Tasks**.

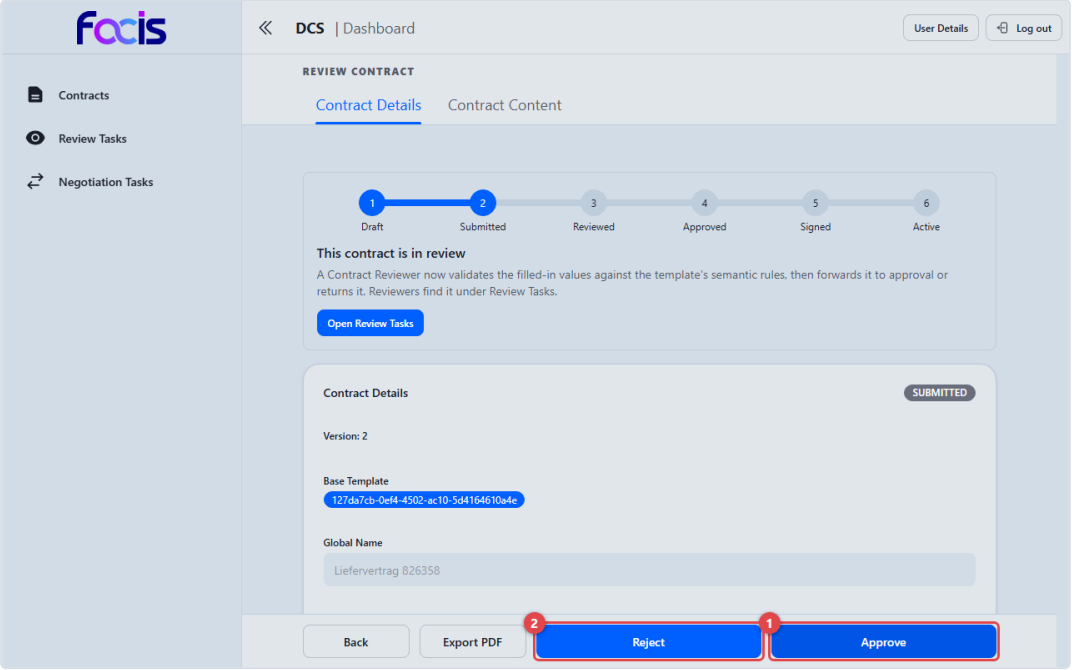
Finding open review tasks

Open **Review Tasks** in the side navigation.



The list shows the contracts in the SUBMITTED status waiting for your review, per entry with name, version, creation date, a badge for the kind of task and its state (**OPEN**). Use the search field, **Filter** and **Sort by** to narrow the list; **View** opens the review view. If you also hold the Template Reviewer role, the same list carries the template tasks as well.

Reviewing a contract and forwarding it



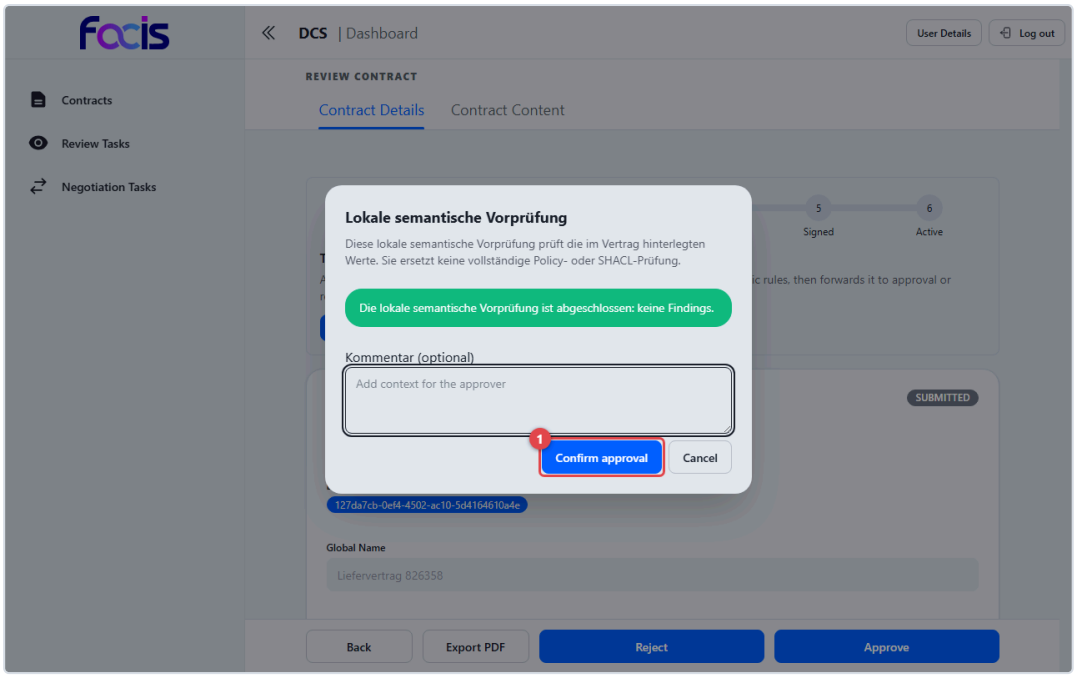
The **Review Contract** view shows the contract read-only with the progress bar (station **Submitted**), the note "This contract is in review" and the tabs **Contract Details** and **Contract Content**. Under **Contract Content** you read the complete contract document with all filled-in values; below it the machine-readable view can be unfolded.

1. **Approve (1)** reviews the contract and forwards it to the Contract Approver for approval (status SUBMITTED → REVIEWED).
2. **Reject (2)** returns the contract to negotiation (status SUBMITTED → NEGOTIATION). The **Comment findings** dialog takes your remarks; text is optional, **Submit** carries out the return. The remark is stored permanently in the contract's audit trail together with the return.

Export PDF and **Back** are also available.

The semantic pre-check

Approve first starts a pre-check of the values stored in the contract. The result appears collected in a dialog so you can judge it before forwarding.



- **No findings:** The **Lokale semantische Vorprüfung** dialog reports "Die lokale semantische Vorprüfung ist abgeschlossen: keine Findings." (the local semantic pre-check completed with no findings). Under **Kommentar (optional)** you can enter a note for the approving person if needed; **Confirm approval** forwards the contract.
- **Findings found:** The dialog lists all found items and states their number; confirmation is not available in that case. Have the items fixed in the contract and start the approval again.
- **Pre-check failed technically:** The dialog shows an error and offers **Retry verification**. The contract was not forwarded.
- **Forwarding failed:** The pre-check succeeded, but the transmission failed. Choose **Retry submission**; your comment is kept in the dialog.

With **Cancel** or the Escape key you close the dialog without submitting a decision.

The pre-check is an early aid. It does not replace the full policy and structure checks that remain authoritative on the server side when the contract is forwarded (see Troubleshooting and FAQ).

When Approve is not available

If **Approve** is not available, check that you are actually signed in with the Contract Reviewer role and that the contract is still waiting for review. A contract that has already been forwarded cannot be reviewed again.

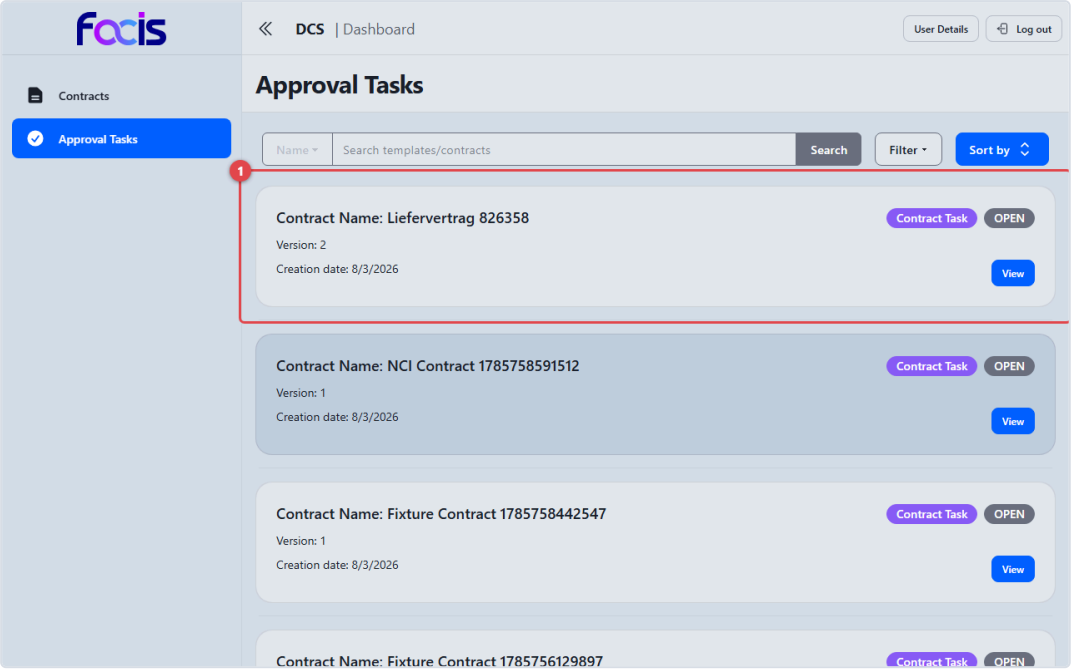
Approving Contracts (Contract Approver)

As a Contract Approver you release reviewed contracts for signing. Only your approval (status APPROVED) opens the contract for signing. Before that, the system allows no signature, and the contract appears in no signing list.

Prerequisites: You are signed in with the Contract Approver role. The side navigation shows **Contracts** and **Approval Tasks**.

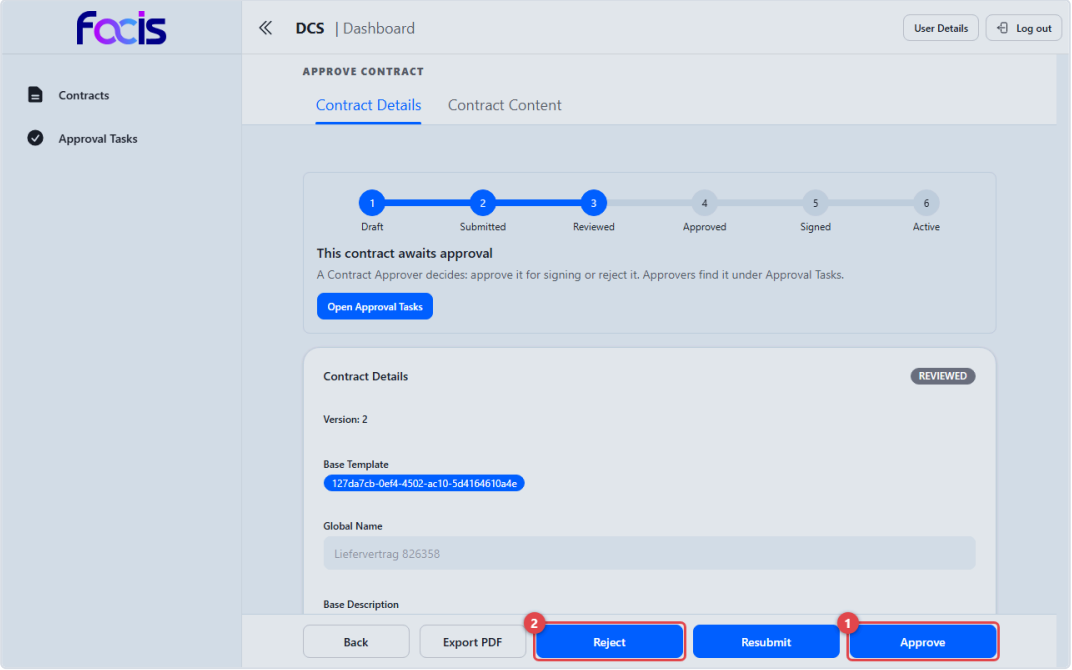
Finding open approvals

Open **Approval Tasks** in the side navigation.



The list shows the contracts in the REVIEWED status waiting for your decision, with search field, **Filter** and sorting. **View** opens the approval view.

Approving or rejecting a contract



The **Approve Contract** view shows the contract read-only with the progress bar (station **Reviewed**) and the note "This contract awaits approval". Under **Contract Content** you read the complete document, under **Contract Details** the master data. The contract's place in a contract hierarchy is shown by the **Structure** tab of the detail view (**Contracts** → **View**).

1. **Approve (1)** releases the contract for signing (status REVIEWED → APPROVED). The contract then appears in the Contract Signers' signing list.
2. **Reject (2)** declines the approval.

Next to them are **Resubmit** (return the contract for another review), **Export PDF** and **Back**.

Each of the three decisions opens a dialog:

- **Approve** only asks for confirmation ("Confirm approval") and is carried out with **Confirm**.
- **Resubmit** offers an optional note under "Add decision note" and is carried out with **Submit**.
- **Reject** requires a reason under "Add rejection reason". Without text nothing is carried out, and the view points out that a reason is required.

Cancel aborts any of these dialogs without consequences.

If a contract is negotiated between two organizations, each side approves its own contract state through its own roles. One organization's approval does not replace the other's.

After approval

The approved contract moves on to signing (see Signing Contracts). Once it is signed, the Contract Manager takes over activation and administration (see Managing Contracts).

Before approving, check whether the contract's target system is set: a contract without a target system is not deployed after signing completes. The Contract Manager makes the selection in the contract view.

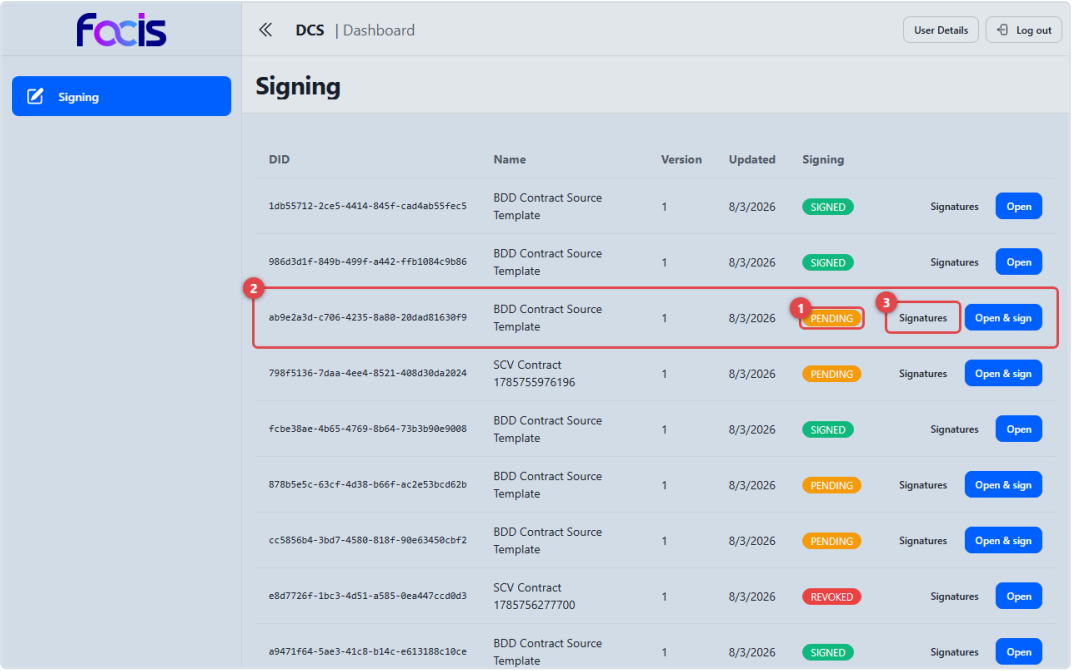
Signing Contracts (Contract Signer)

As a Contract Signer you sign approved contracts. The DCS guides you step by step through the **Secure Contract Viewer**: you check the contract, prove your identity and your signing authority with your wallet, download the document to sign, sign it with your **own** signing key outside the DCS and upload the signed document again. The DCS itself keeps no signing key for you.

Prerequisites: You are signed in with the Contract Signer role and hold a wallet with your identity credential (PID) and your power of attorney (PoA), plus your own signing tool. The side navigation shows **Signing**.

The signing list

Open **Signing** in the side navigation.



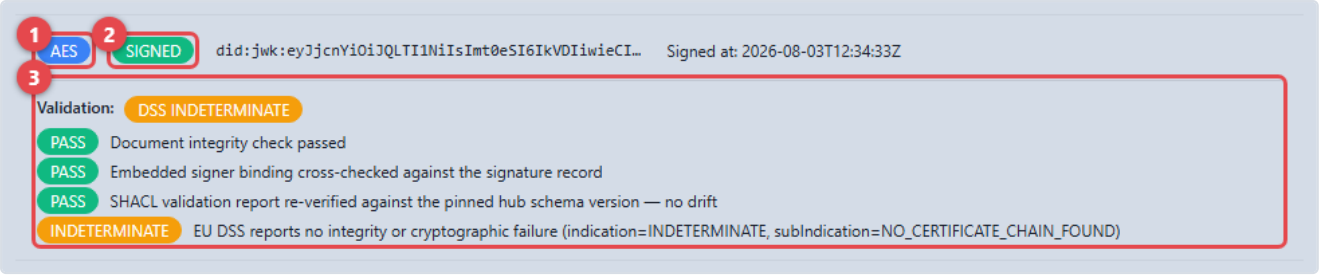
The list shows per contract the columns **DID**, **Name**, **Version**, **Updated** and **Signing**. The signing status **(1)** in the **Signing** column tells you what to do; the corresponding row **(2)** carries **Signatures (3)** on the right and the matching main action:

- **PENDING:** the contract is approved for signing and waits for your signature. **Open & sign** opens it in the Secure Contract Viewer.
- **SIGNED:** the contract is signed and remains listed as signed after activation as well. The button here is called **Open**.
- **REVOKED:** an applied signature was revoked. Revoked contracts also remain visible in the list so the process stays traceable.

Signatures (3) unfolds the signature details per row; the same button is then called **Hide signatures** and folds them away again.

A contract that is not yet approved does not appear here at all and consequently cannot be signed. If there are no contracts, the view shows the note "Nothing awaits your signature. Contracts appear here once they are approved for signing."

Inspecting signature details

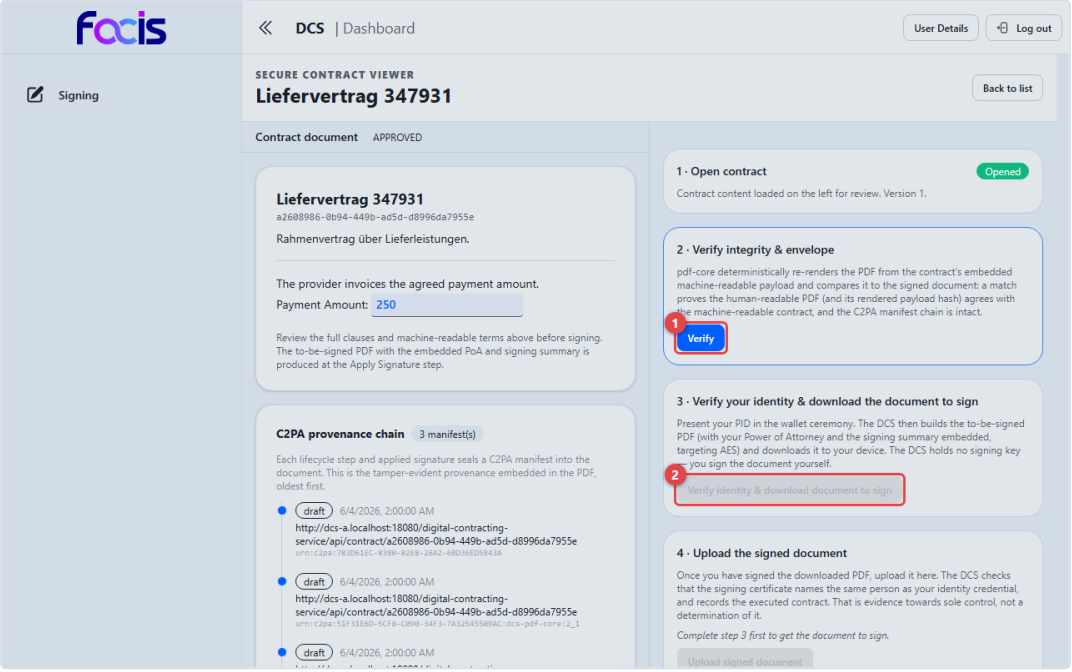


For each applied signature you see:

- the proven signature level or the credential used **(1)** (in the example **AES**); if none was established, it explicitly reads **NOT ESTABLISHED**,
- the status of the signature **(2)** (**SIGNED** or **REVOKED**),
- the identifier of the signatory,
- the timestamps ("Signed at: …", for a revocation additionally "Revoked at: …").

Below that, the **Validation** section **(3)** summarizes the check results: the result of the signature validation under the EU procedure and the individual integrity findings with their verdict (**PASS**, **FAIL** or **INDETERMINATE**). If no validation result exists yet, it reads "No validation result available yet." If no signature has been applied yet, the note "No signature has been applied yet." appears.

The Secure Contract Viewer



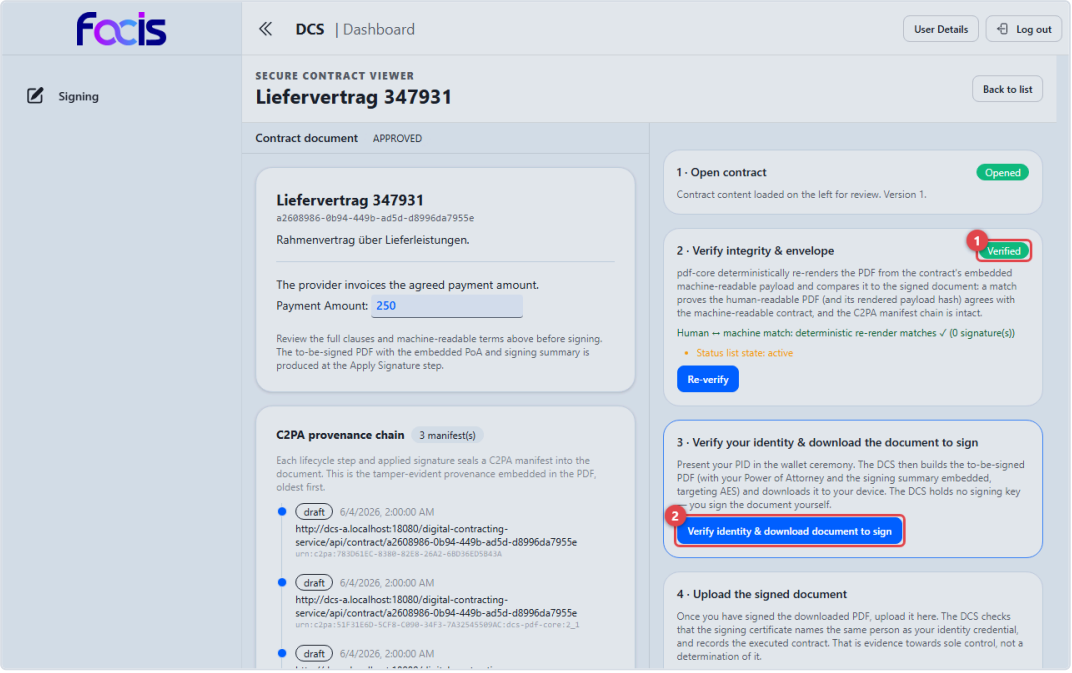
On the left, under **Contract document**, you read the complete contract document with all clauses and the filled-in values; the badge next to the heading shows the contract status. If the document already carries provenance evidence, the **C2PA provenance chain** card sits below it with the history of all previous processing steps, anchored tamper-proof in the document, oldest first. **Back to list** at the top right returns to the signing list.

On the right, five steps guide you through signing, each with its own badge for its state:

- **1 · Open contract:** the contract content is loaded (badge **Opened**), the version number is shown below.
- **2 · Verify integrity & envelope:** with **Verify (1)** you check that the readable document exactly matches the machine-readable contract content and that the provenance chain is intact.
- **3 · Verify your identity & download the document to sign:** the button **(2)** is locked until step 2 has succeeded. The text above it states that the DCS holds no signing key: you sign yourself.
- **4 · Upload the signed document:** locked as well; the note "Complete step 3 first to get the document to sign." explains the order.
- **5 · Validate applied signatures:** after the upload, checks the applied signatures against the stored trust requirements. As long as nothing is uploaded, **Validate** stays locked.

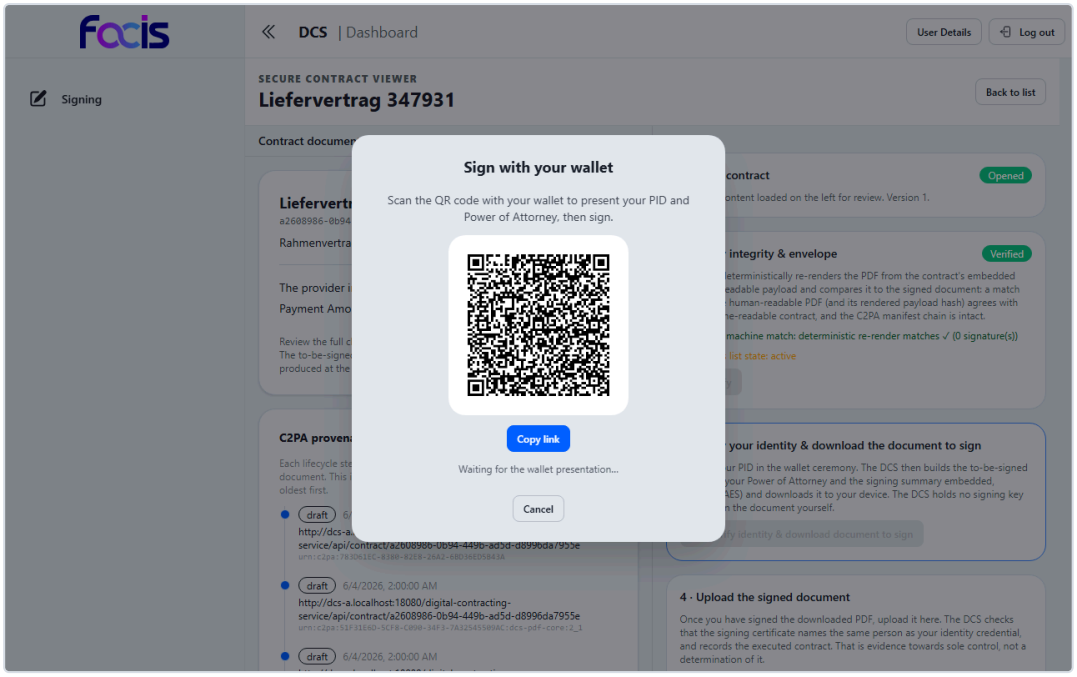
Signing step by step

1. Check the contract. Click **Verify**.



After successful verification, step 2 shows the **Verified** badge **(1)** and below it the result of the match between readable document and machine-readable contract content ("Human ↔ machine match: deterministic re-render matches") plus the queried revocation status. Only then is the button in step 3 **(2)** unlocked; the upload area in step 4 stays locked until the document to sign is available. **Re-verify** repeats the check at any time. If the match fails, step 2 carries the **Not verified** badge.

2. Prove identity and authority. Click **Verify identity & download document to sign**. The wallet dialog opens.



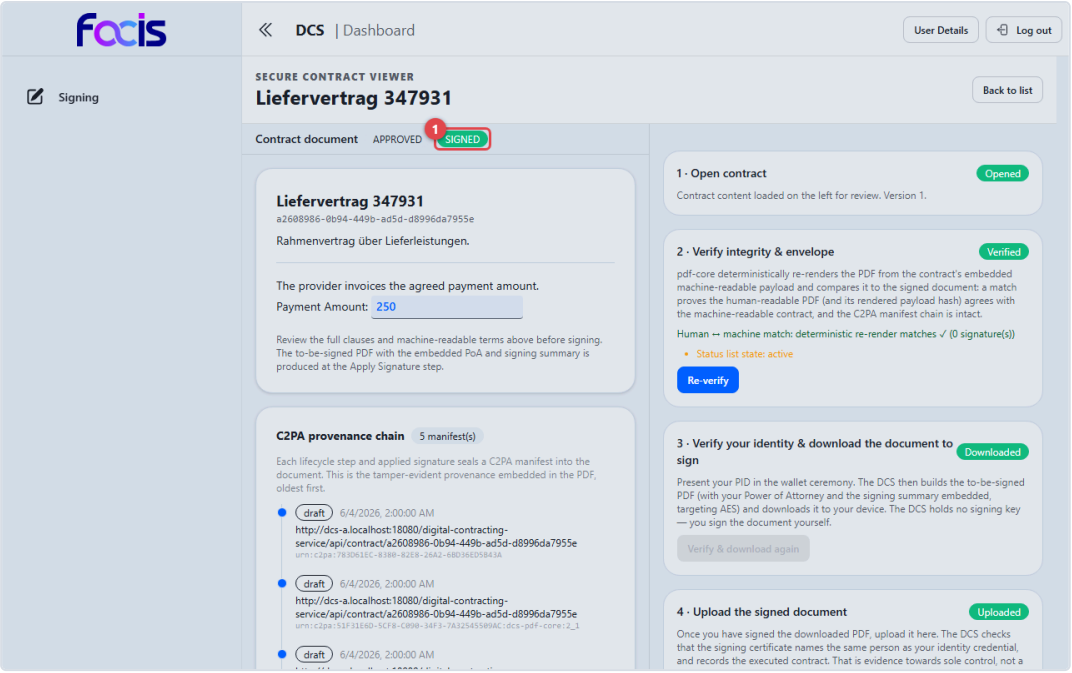
The **Sign with your wallet** dialog asks you to scan the QR code with your wallet and present your identity credential and power of attorney there. Alternatively, **Copy link** puts the same link on the clipboard. The dialog waits ("Waiting for the wallet presentation...") until the wallet has confirmed; **Cancel** aborts.

The DCS checks whether the authority is issued to you personally, whether it names the signing organization, whether it is currently valid and whether it traces back to a trust anchor accepted for your environment. If any of these points is missing, the process stops. For a contract with a partner organization, their system repeats this check before accepting your signature.

The DCS then prepares the PDF to sign, with the embedded authority and signature overview, and downloads it to your device. Step 3 then carries the **Downloaded** badge, and the button is called **Verify & download again**; it fetches the same file once more.

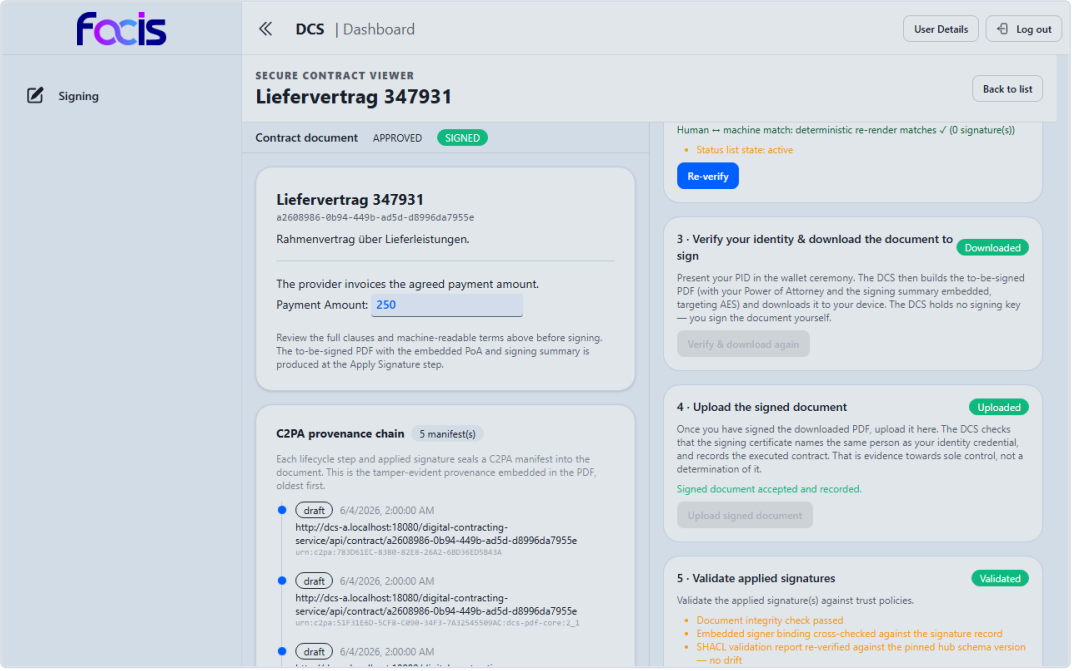
3. Sign the document. Sign the downloaded PDF with your own signing tool outside the DCS. The document already contains the signature field intended for you.

4. Upload the signed document. In step 4 choose **Upload signed document** and upload the signed PDF. The DCS checks that the signing certificate names the same person as your identity credential and sets the contract to **SIGNED**.



Next to the **Contract document** heading the **SIGNED** badge **(1)** appears. Step 3 stands at **Downloaded**, step 4 at **Uploaded**, and the provenance chain on the left has gained two more entries. In step 5, **Validate (2)** is now usable.

5. Validate the signatures. Validate in step 5 checks the applied signatures. Without findings, the view reports "Signature validation passed with no findings."; otherwise it lists the findings. **Re-validate** repeats the check.



After the check, step 5 carries the **Validated** badge and lists the individual findings. Step 4 reports "Signed document accepted and recorded.", and the buttons of the completed steps are locked. On the left, the **C2PA provenance chain** grows by the entries that sealed the preparation and the signature.

The contract then carries the status **SIGNED**. For a contract with a partner organization, the signed document is transmitted to the counterparty automatically; their signatory applies their signature on the same document as well. Only when all intended signatures are verifiably present does the contract count as fully signed.

Contracts with several signature fields

A contract can provide several signature fields, for example for two authorized signatories. In that case:

- Each field is signed by the person intended for it, one field after another. A field that is already signed cannot be signed a second time.
- The first signature is only accepted once **all** intended signatories have proven their identity (step 3). So coordinate first that everyone has completed this step before the first person uploads their signed document.
- The contract is only activated once all intended fields are signed. An attempt to deploy it earlier is rejected.

If you hold the Contract Manager role instead of Contract Signer, opening, verifying and validating are available, but signing itself is not. The view points this out. A Contract Observer can also open the view but only read: **Verify** and **Validate** stay locked for them, and a note explains that verification, validation and signing are reserved for a Signer or Contract Manager.

When a signature is rejected

The DCS accepts only signatures that beyond doubt come from the person who previously identified themselves with the wallet. These cases can come your way:

- **The signing certificate does not match your identity credential.** If the certificate the PDF was signed with names someone other than the person proven in the wallet, the DCS rejects the upload. The view names the concrete reason (the certificate "does not identify the verified signatory"); the contract is not set to **SIGNED**. Use a signing tool whose certificate is issued in your own name and upload again.
- **The uploaded document is not the one provided.** If the content of the signed document differs from what the DCS provided for signing, the upload is rejected. Sign exactly the downloaded file and do not change it in the process.
- **The signature level is not sufficient.** If the contract requires a higher signature level for your signature field (for example a qualified signature, QES), a signature of a lower level is rejected. Use a signing tool of the required level.
- **The counterparty has not yet confirmed this version.** For a contract with a partner organization, the DCS only allows your signature once the counterparty has bindingly confirmed the same contract state by closing its own negotiation round. Until then the view reports "Waiting for the counterparty to settle this version …". This is not a rejection of you personally: there is nothing you need to do, signing opens as soon as the counterparty's confirmation arrives. If renegotiation happened in the meantime, the counterparty must confirm the new version again.
- **Your identity credential is revoked.** A revoked identity credential is rejected already at presentation in the wallet; the signing process does not continue. Have a valid credential issued.

- **The authority does not trace back to an accepted trust anchor.** The authority is rejected, the signing process remains unverified. Contact the body that issued your authority.
- **The authority is revoked or suspended.** The signature does not proceed. Have a valid authority issued.
- **Authority without a checkable status proof.** If the status proof is missing, unknown or invalid, the DCS rejects the check for safety reasons.
- **The deadline of the signing process has passed.** A time limit applies to every signing process. If your wallet confirms the identity proof only after it has passed, the presentation is rejected. Restart signing via **Open & sign**.

Visible safeguards

- The steps are unlocked strictly one after another; skipping is not possible.
- A contract that is not approved does not show up in the signing list in the first place.
- If the integrity check fails, the download stays locked. Only what demonstrably matches the negotiated content gets signed.
- If an uploaded signature is rejected, the view always names the concrete reason; the contract stays unsigned.

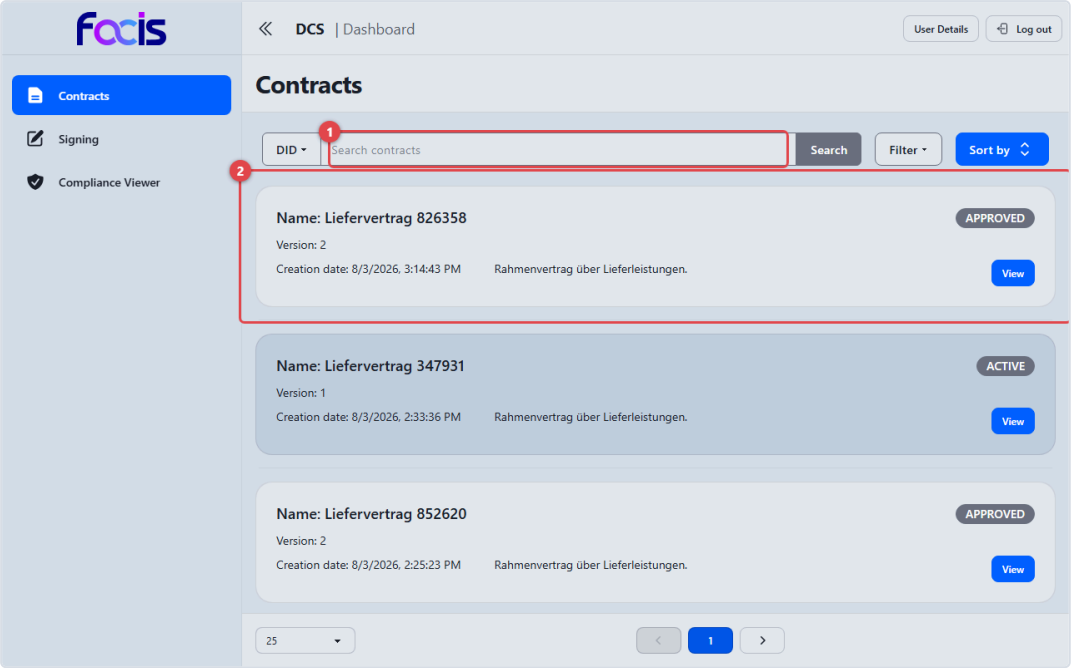
Managing Contracts (Contract Manager)

As a Contract Manager you keep the whole contract portfolio in view, export contract documents, decide which target system a contract is handed over to on activation, activate fully signed contracts, validate signatures in the Compliance Viewer and terminate contracts.

Prerequisites: You are signed in with the Contract Manager role. The side navigation shows **Contracts**, **Signing** and **Compliance Viewer**.

The contract overview

Open **Contracts** in the side navigation.



1. Use the search field **(1)** to find a contract. The selector in front of it determines what is searched; the default is the identifier (**DID**), the other options are **Name**, **Description**, **Version** and **Contract Data**. **Search** runs the search, **Filter** narrows by status (for example only **DRAFT** or only **APPROVED**).
2. Each entry **(2)** shows the contract name, version, creation date and the status badge (**DRAFT**, **OFFERED**, **REJECTED**, **WITHDRAWN**, **NEGOTIATION**, **SUBMITTED**, **REVIEWED**, **APPROVED**, **SIGNED**, **ACTIVE**, **REVOKED**, **TERMINATED**, **EXPIRED**). **View** opens the detail view.

If a contract carries an expiration date, it is shown in the entry. Once the remaining term falls below the agreed notice period, a coloured badge with the remaining time appears next to it ("Contract expires in ..."). Two more badges concern the underlying template: "A newer template version is available" and "This contract uses a deprecated template". Both are also links to the template.

Contract details and PDF export

The detail view (**View**) shows the contract with the tabs **Contract Details**, **Contract Content** and **Structure**. Under **Contract Content** you see the complete readable document generated from the machine-readable contract data; below it the **Machine-readable view** area unfolds the same content in field form: **Field** (label), **Value** and **Identifier** (technical identifier). Both representations come from the same contract content and therefore cannot differ. Automatic checks and the KPIs reported by the target system refer to the identifiers: a reported KPI names the field by its identifier, not by its label.

The **Contract Details** tab lists, besides name and description, the contract deadlines: **Expiration Notice Period (in days)** and **Expiration Policy** (**Renewal**, **Archiving** or **Termination**). For you these fields are read-only; they are set at creation and during negotiation.

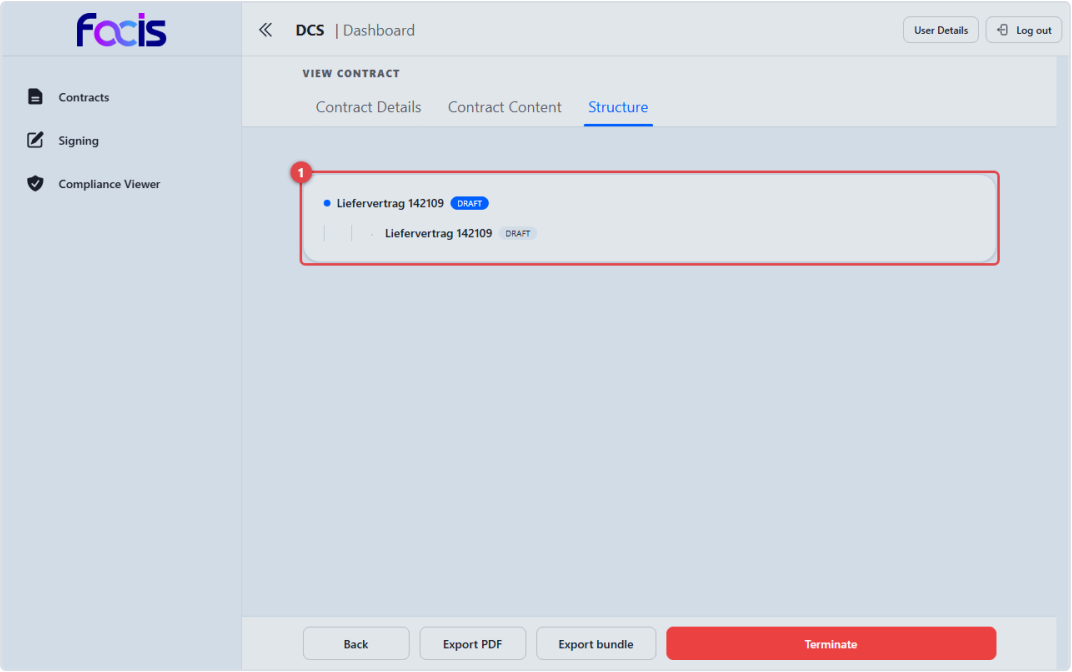
- **Export PDF** downloads the current contract document as an archive PDF. The PDF carries the machine-readable contract content and the complete provenance chain within itself. It is the same document that is exchanged between the systems for cross-organization contracts. Immediately after a change, the export may wait briefly until the document has been regenerated.
- **Export bundle** produces a complete package of the contract with all related contracts of the hierarchy known to this installation. The package is only produced complete: if a part is missing, for example because no contract document has been generated yet for one of the contracts, the export is rejected and a red message names the reason.

Contract hierarchy (the "Structure" tab)

A contract can be created as a sub-contract of a parent contract (the Contract Creator makes the assignment at creation, see Creating and Negotiating Contracts). The detail view shows these relationships in two places:

- In the **Contract Details** tab, the **Part of Contract** card names the parent contract and the **Component Contracts** card the assigned sub-contracts, each as a link to their detail view.
- The **Structure** tab presents the same assignment as a tree. A contract without an assignment reports "This contract has no parent or child contracts." there.

The **Component Contracts** card **(1)** in the **Contract Details** tab of a parent contract: each sub-contract as a clickable badge.

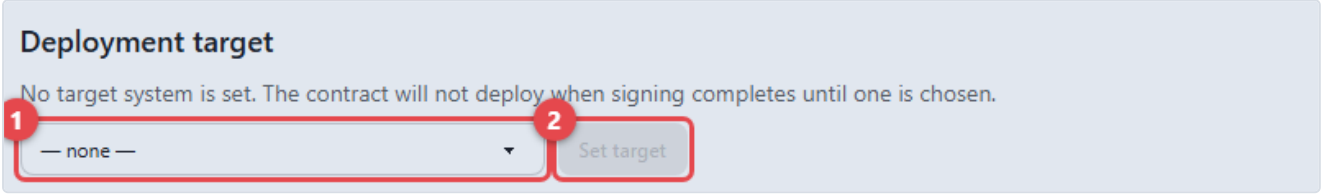


In the tree **(1)** the opened contract is marked with a dot and its status badge; above it are the parent contracts, indented below it the sub-contracts, each with its own status badge and as a link into its detail view.

Every contract has at most one parent contract. **Export bundle** includes the contracts of the hierarchy known to this installation in the complete package.

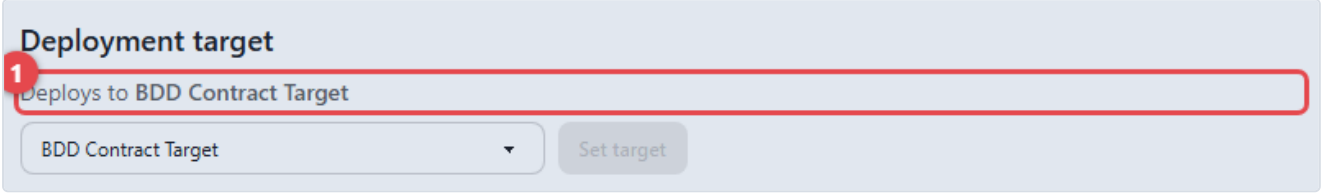
Setting a contract's target system

Each contract itself names the target system it is handed over to on activation. You make the selection in the detail view, in the **Contract Details** tab, in the **Deployment target** card.



1. In the selector **(1)** choose one of the registered target systems. The list comes from the target system administration (see System Administration); the top entry ("— none —") means no target.
2. **Set target (2)** saves the selection. The button stays locked while the selection is unchanged.

Set the target **before signing**. When signing completes, the contract is handed over to its target system automatically; a contract without a set target is not deployed. The card says so explicitly: "No target system is set. The contract will not deploy when signing completes until one is chosen."

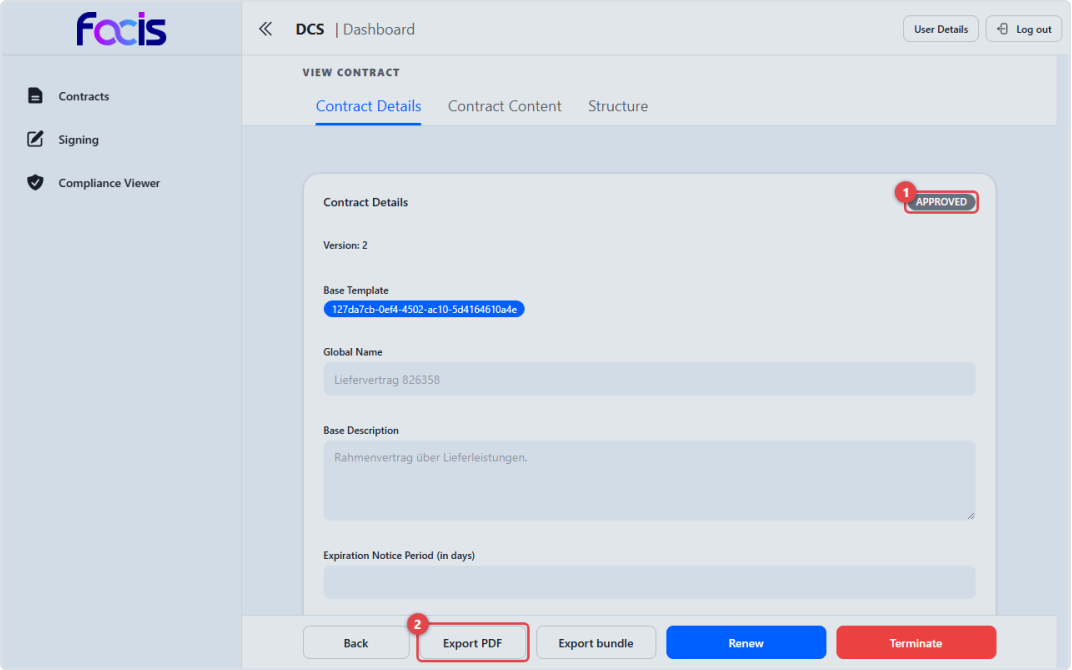


After saving, the view reloads the contract and shows the set target as "Deploys to *Name*". The target stays changeable as long as the contract is not terminated, expired or revoked; a wrong selection can therefore be corrected up to signing. If saving fails, a red error message appears directly in the card.

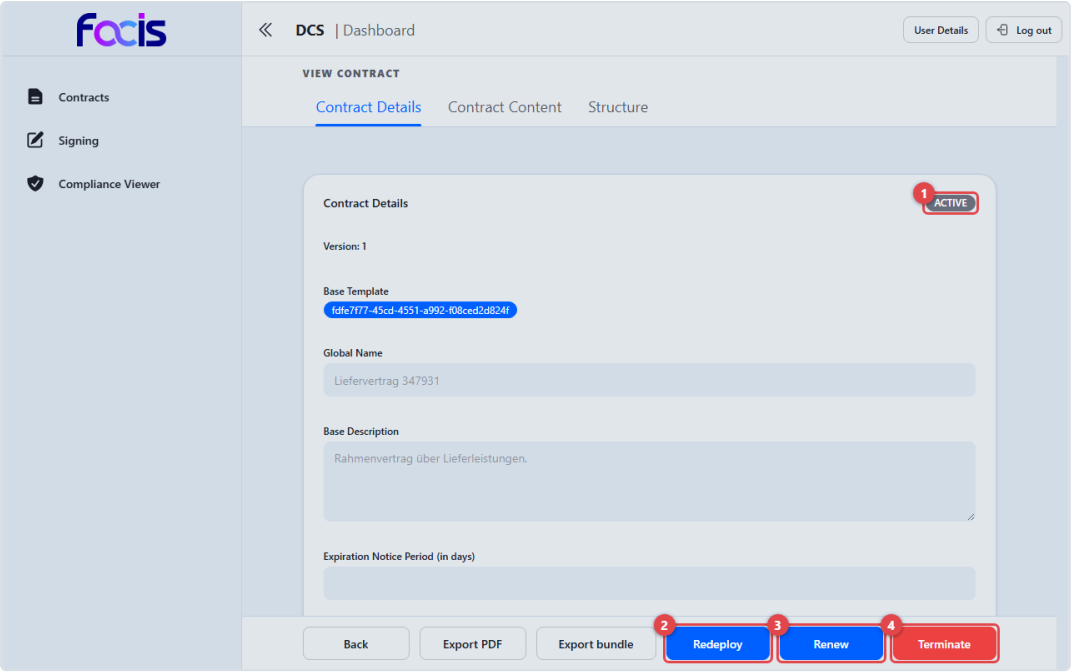
Activating a signed contract

Activation hands the contract over to its set target system, which measures the agreed KPIs against the machine-readable contract rules. It happens in two ways:

- **Automatically:** As soon as all intended signatures are verifiably present, the contract is transmitted to its target system without further action. Once the target system confirms receipt, the status changes from **SIGNED** to **ACTIVE**.
- **Manually:** For a contract in the **SIGNED** status, the detail view offers the **Deploy** action. A contract that is not yet fully signed is rejected for activation.



The detail view of an approved contract: the status badge **(1)** and the action bar with **Back**, **Export PDF (2)**, **Export bundle**, **Renew** and **Terminate**. **Terminate** is available to you in every state as long as the contract is not already terminated. On an offered or negotiating contract, **Review offer** or **Negotiate** additionally appears: with it you open the negotiation view as Contract Manager, for example to handle an offer received from a partner organization when your installation has not staffed a dedicated negotiation role (see Creating and Negotiating Contracts).



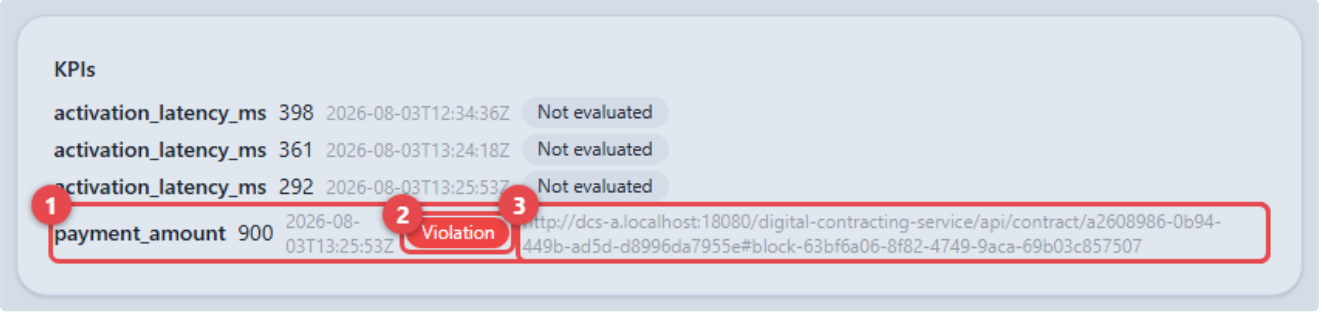
After signing, the same bar looks different. Here the target system has already confirmed receipt, the contract stands at **ACTIVE (1)**, and the deployment action is therefore called **Redeploy (2)**. For a contract in the SIGNED status it reads **Deploy**. Next to it sit **Renew (3)** and **Terminate (4)**.

Deploying again (Redeploy)

On an active contract the action is called **Redeploy**. It transmits the contract to its target system again, for example when the target system was temporarily unreachable or needs to receive the contract again. The contract status stays **ACTIVE**.

Reported KPIs

If the target system reports KPIs for an active contract, they appear in the detail view in the **Contract Details** tab in the **KPIs** area.



1. Each row **(1)** shows the name of the KPI, the reported value and the reporting time.
2. The badge **(2)** shows the target system's verdict: **Violation** (red) for a breach of a limit agreed in the contract, **Satisfied** for compliance and **Not evaluated** when the target system gave no verdict.
3. For an evaluated KPI, the identifier of the rule **(3)** the verdict refers to follows behind it. This lets you trace the breach back to the individual contract clause.

The verdict comes from the target system that executes the contract and observes the measured values; the DCS records it and does not derive it itself. A breach additionally appears in compliance monitoring (see Monitoring Compliance).

Creating a renewal contract (Renew)

Renew creates a follow-up contract for an existing contract, for example when an expiring contract is to be continued. The action is available from approval onward (states **APPROVED**, **SIGNED**, **ACTIVE**, **TERMINATED** and **EXPIRED**). The confirmation dialog asks "Create a renewal contract from this one? The original is left untouched."

After the confirmation, a **new, standalone contract** in the **DRAFT** status comes into being with its own term data and a fixed reference to the original contract; the view switches to its detail view. The original contract remains unchanged in its previous state. The new contract then passes through the usual path of negotiation, review, approval and signing.

The **Renew** button sits in the action bar of the detail view, between **Export bundle** and **Terminate** (see the two figures above).

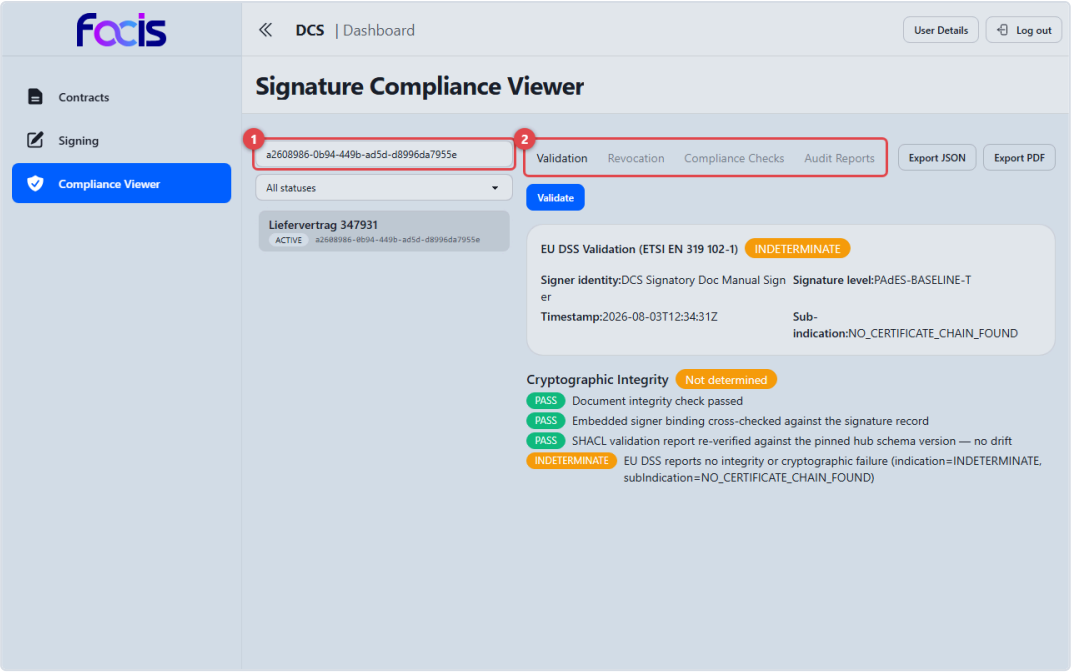
Terminating a contract

Terminate ends a contract. The action does not require a signature: even a contract that has only been approved can be terminated, for example when the undertaking is abandoned before signing. The confirmation dialog asks "Proceed with terminating?" and requires a **Reason**; without a reason, **Submit** has no effect. Reason, acting person and time are stored permanently in the contract's audit trail.

After terminating, the view returns to the contract overview; the contract carries the status **TERMINATED** there and in the detail view. It stays readable and exportable but can no longer be changed and cannot be terminated a second time. A renewal contract can still be created via **Renew**.

Signature Compliance Viewer

The **Compliance Viewer** area validates the signatures of signed contracts under the European validation standards.

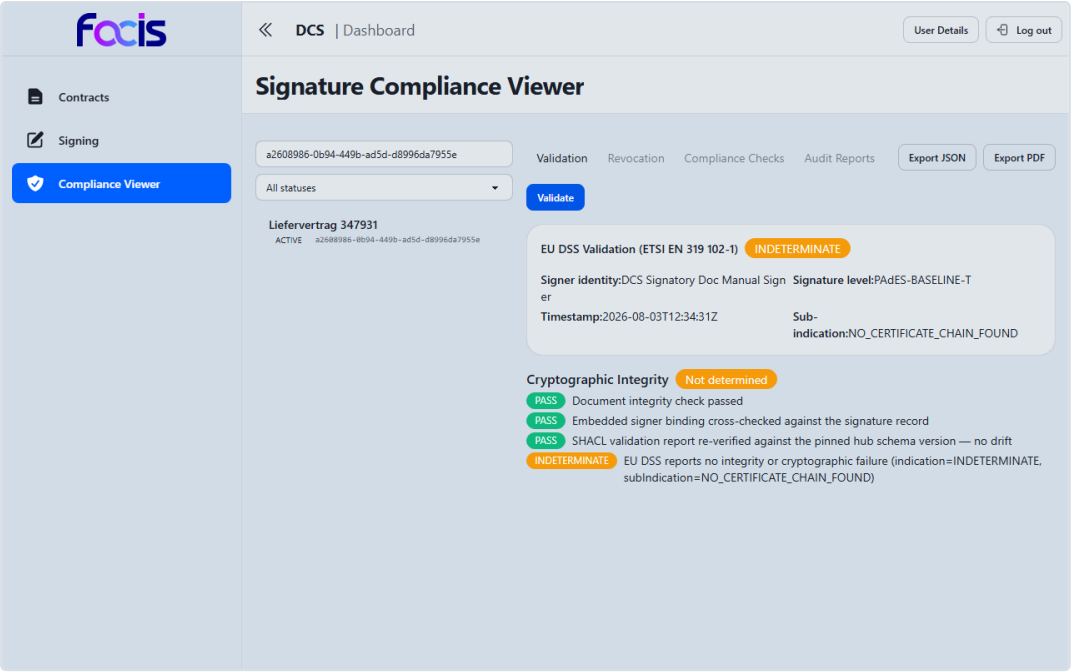


On the left, the view lists all contracts with their status; use the search field **(1)** to find a contract by name or identifier, and the selector next to it narrows the list to one status. Clicking an entry opens the validation view on the right with four tabs **(2)**:

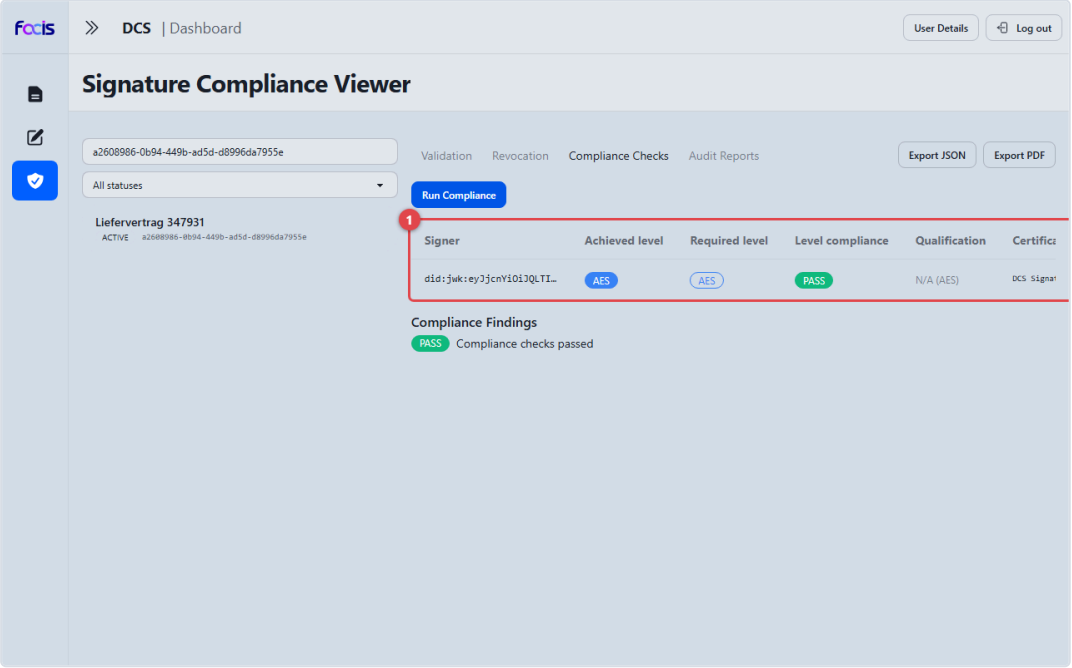
- **Validation: Validate** starts the signature validation under the EU validation procedure.
- **Revocation:** lists the applied signatures with their status and allows revocation.
- **Compliance Checks: Run Compliance** rates, per signature, the achieved against the required signature level and the status of the underlying credential.
- **Audit Reports: Load Audit Report** loads the audit history of the signatures (see Running Audits).

To the right of the tabs sit **Export JSON** and **Export PDF**. Both become active once the signature data of the chosen contract is loaded, and they are locked while a revocation is running.

While data is loading, the view says so ("Loading contracts...", "Loading signature data..."). An empty list is never shown as "no matches" before loading has happened: without signed contracts it reads "No signed contracts are available.", with too narrow a restriction "No contracts match the current filters."



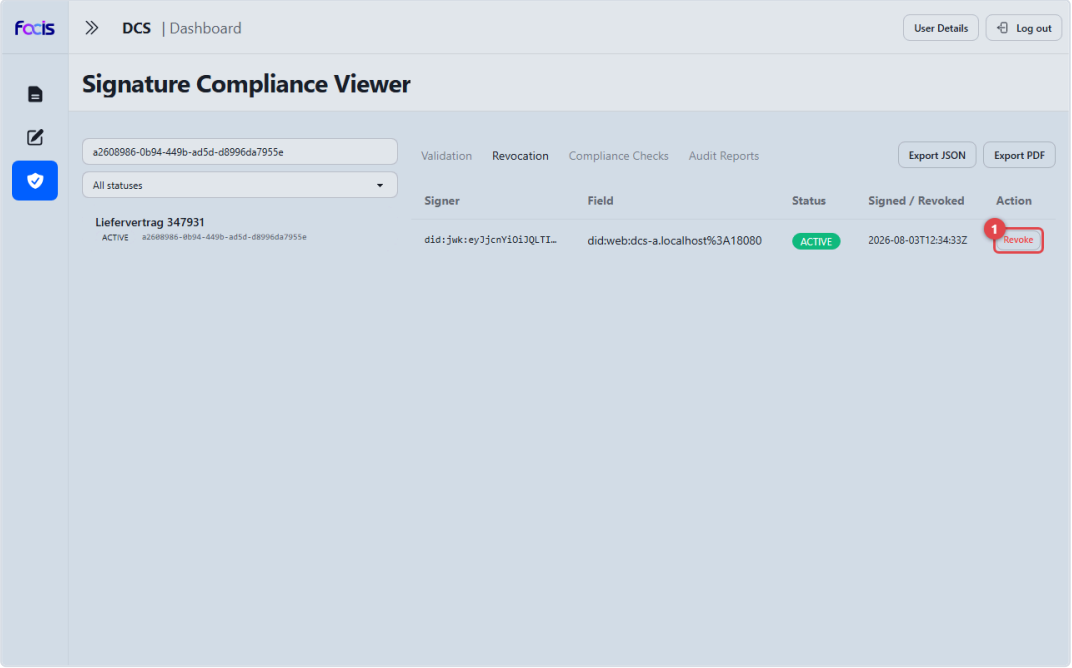
The **Validation** tab after **Validate**. The **EU DSS Validation (ETSI EN 319 102-1)** card carries the overall verdict of the validation procedure at the top right and names below it the identity of the signatory (**Signer identity**), the signature level (**Signature level**, here PAdES-BASELINE-T), the timestamp (**Timestamp**) and the finer classification (**Sub-indication**). Below that, **Cryptographic Integrity** shows the overall verdict with the individual findings, each with its verdict **PASS**, **FAIL** or **INDETERMINATE**. The example shows a signature whose content is intact (three passed findings), while the EU procedure reaches no final verdict because it lacks the signatory's certificate chain (**NO_CERTIFICATE_CHAIN_FOUND**). If the contract carries no signature, the check reports "No signatures found for the contract".



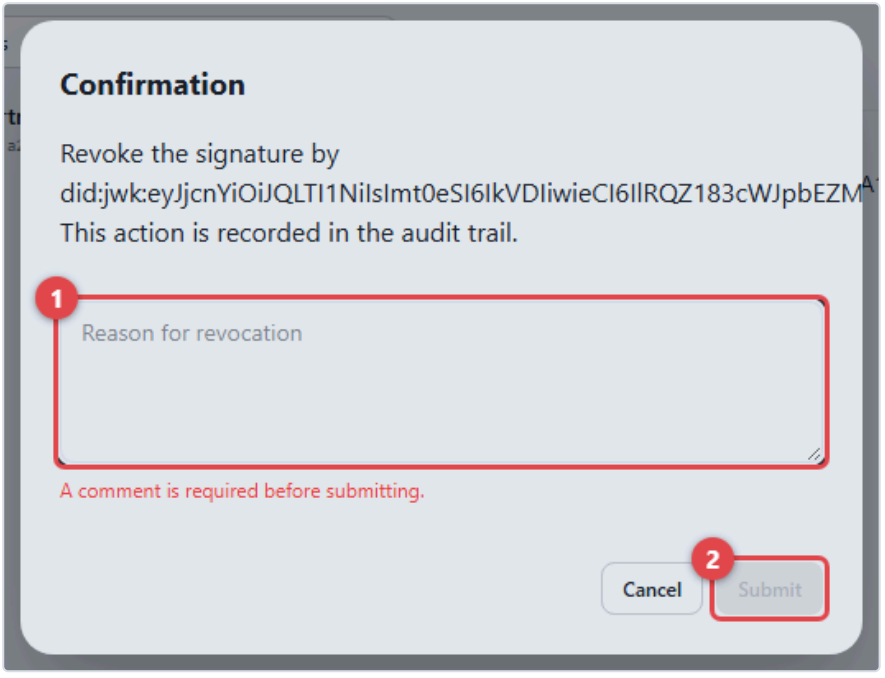
The **Compliance Checks** tab after **Run Compliance**: a table **(1)** with one row per signature. It lists **Signer**, **Achieved level**, **Required level**, **Level compliance**, **Qualification**, **Certificate subject**, **Credential status** and **Credential binding**; the table is wider than the screen and continues to the right. In the example the

signature reaches the required level (**AES** against **AES**, **PASS**); **Qualification** stands at **N/A (AES)** because qualification is only judged for a qualified signature. Below, **Compliance Findings** summarizes the result ("Compliance checks passed"). If the contract carries no signature, it reads "No signatures on this contract."

Revoking a signature



The **Revocation** tab lists the applied signatures with **Signer**, **Field**, **Status**, **Signed / Revoked** and the **Action** column; if the contract carries no signature, a corresponding note is shown. Via **Revoke (1)** in the **Action** column you revoke an applied signature.



The confirmation dialog names the affected signatory and requires a reason under **Reason for revocation (1)**. **Submit (2)** carries out the revocation, **Cancel** aborts without consequences. While the field is empty, **Submit** stays locked. A signature that is already revoked cannot be revoked a second time.

After the revocation the signature stands at **REVOKED**. The contract itself is also carried as **REVOKED**; it remains visible in the signatories' signing list, there with the revocation time in the signature details. For a contract with a partner organization, the revocation is transmitted to the counterparty automatically.

Revocation requires the Contract Manager role. Without it, **Revoke** stays locked, and below the table it reads "Revocation requires the Contract Manager role."

Telling lifecycle and revocation status apart

When verifying an exported contract, two pieces of information are reported separately:

- the **lifecycle status** of the contract, for example `active`, `suspended` or `terminated`;
- the **queried revocation status** (`active` or `revoked`) together with the result of that query (`passed` / `failed`) and, if applicable, an error reason.

This keeps "terminated" visible as the contract state while the revocation status can report "revoked" at the same time. A confirmed revocation or suspension is taken into account within a few minutes on a fresh check. If the status service is unreachable, verification reports the revocation status as `unavailable` and the status check as `failed`; the overall result is then not successful, and the contract is explicitly not reported as active.

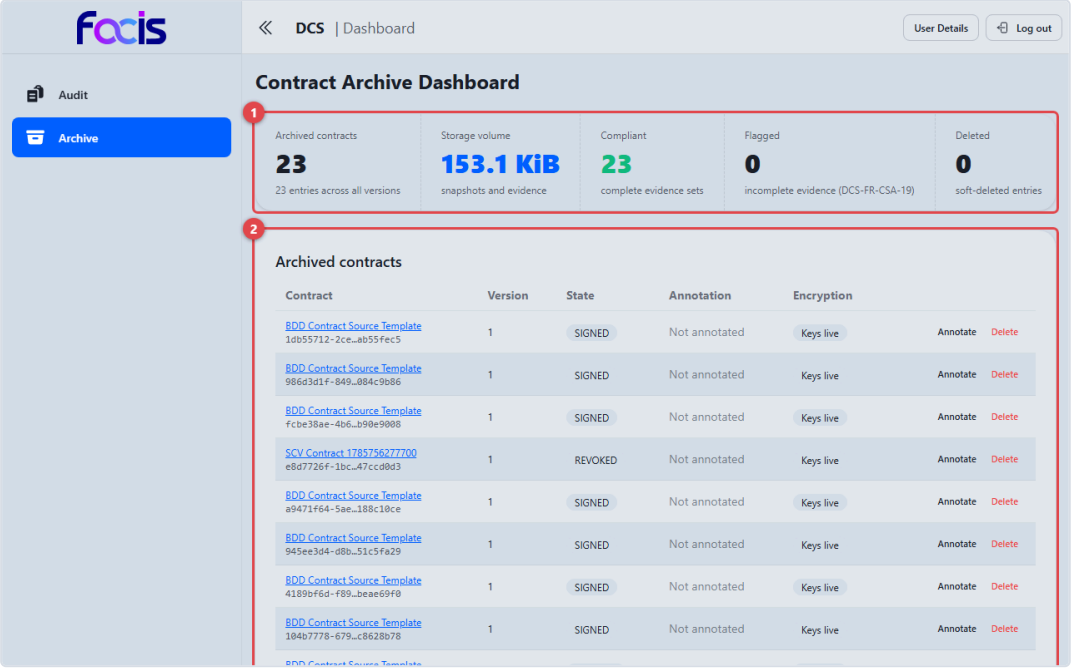
Managing the Archive and Retrieving Contracts (Archive Manager)

As soon as a contract is signed, it is archived automatically, together with a snapshot of the document, the signature details, a checksum of the content and a timestamp proof. As an Archive Manager you oversee this stock, retrieve archived contracts together with their evidence chain and carry out erasure requests.

Prerequisites: You are signed in with the Archive Manager role. The side navigation shows **Archive** and **Audit**. Auditors see the same areas (see Running Audits).

The archive dashboard

Open **Archive** in the side navigation.



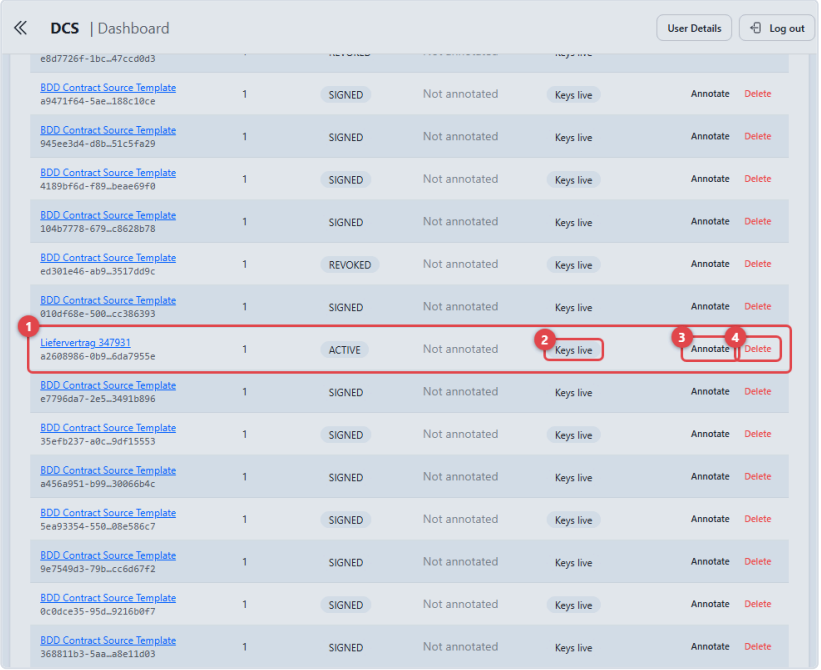
The **Contract Archive Dashboard** view is organized into four areas:

1. The statistics bar **(1)** with **Archived contracts** (number of archived contracts), **Storage volume** (storage used), **Compliant** (entries with complete evidence), **Flagged** (entries with incomplete evidence) and **Deleted** (entries whose content has been erased).
2. **Archived contracts (2)**, the list of archive entries. It is only available to the Archive Manager role. If there is no entry, it reads "No archived contracts.".

Further down, below the section visible in the image, two more areas follow:

- **Expiring within 30 days**, the contracts whose validity ends soon, with name and expiry time. Which period counts as "soon" is configurable on the system side; contracts outside the window are not listed as expiring.
- **Recent actions**, the latest archive operations with time (**When**), operation (**Operation**), triggering identity (**Actor**) and the affected contract (**Contract**) as a link.

The statistics bar in detail. Below each number it says what it refers to, for example "complete evidence sets" below **Compliant**.

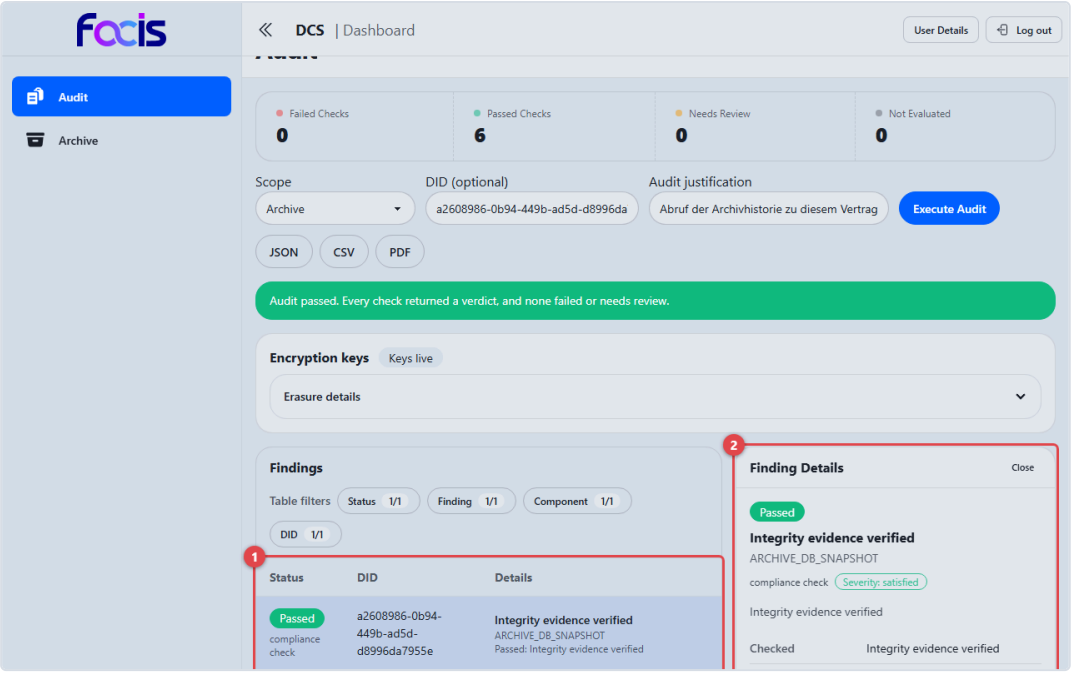


The **Archived contracts** list shows per entry the columns **Contract** (name and identifier, the name is a link), **Version**, **State**, **Annotation** and **Encryption**. The highlighted entry **(1)** shows the interplay: the contract's status badge, the erasure status **Keys live (2)** and, on the right, the actions **Annotate (3)** and **Delete (4)**.

Retrieving an archived contract

An archive entry is retrieved through the audit workspace. There you receive the contract together with its complete, verifiable archive history.

1. In the dashboard, click the name of the contract you want, in the **Archived contracts** list, under **Expiring within 30 days** or under **Recent actions**. The application switches to the audit workspace and sets everything up there already: **Scope** stands at **Archive**, the **DID (optional)** field contains the contract's identifier.
2. Under **Audit justification** enter the justification for your retrieval.
3. **Execute Audit** retrieves the contract's archive history.



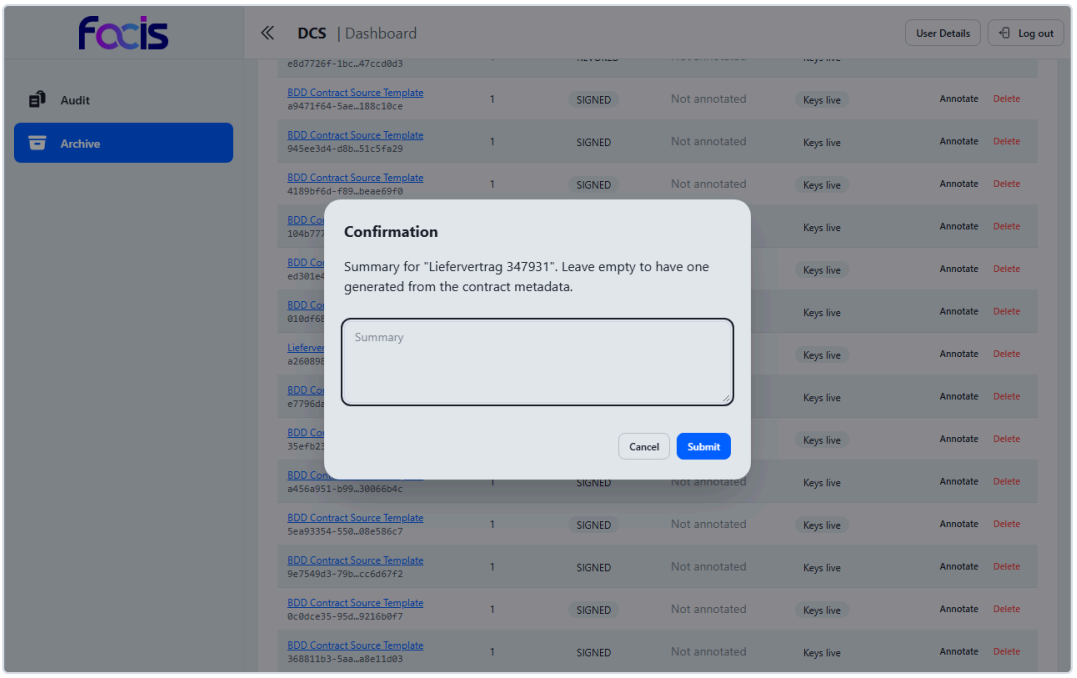
The result appears as the **Findings** list **(1)**: per row one finding about the archive entry with verdict, identifier and short description, in the example the passed proof "Integrity evidence verified" over the archive snapshot. Clicking a row opens the full evidence for it on the right under **Finding Details (2)**, for example the audited statement, the severity and the component involved. Above the list sits the **Encryption keys** section with the entry's erasure status. Via **JSON**, **CSV** and **PDF** you export the evidence; the export requires its own justification.

If you already know the contract's identifier, you can also open the audit workspace directly, set **Scope** to **Archive** and enter the identifier. The route via the dashboard spares you the typing.

Annotating an archive entry (Annotate)

Each archive entry can carry a short description (**Summary**) and tags (**Tags**) to organize the stock by topic. Both are shown in the **Annotation** column; without an annotation it reads "Not annotated". The annotation is the only changeable part of an archive entry; the archived content and evidence themselves stay immutable.

1. In the entry's row choose **Annotate**.



The first dialog names the affected contract and takes the short description. If you leave the field empty, the system itself generates a summary from the contract's master data (name, version, state, creator). **Submit** accepts the input, **Cancel** aborts.

2. The second dialog takes the tags, separated by commas (for example "supply, 2026, renewed"). Without input, the existing tags are kept; input replaces them completely.

The annotation then appears in the **Annotation** column, the tags as small badges below it. Every annotation is recorded in the archive log. Reading roles cannot annotate; such an attempt is rejected.

Known limitation: The second dialog for the tags currently does not appear reliably. If it stays away, the annotation is not saved, and the **Annotation** column keeps showing "Not annotated". Report this to the operations team of your installation; the annotation can be added there through the system interface.

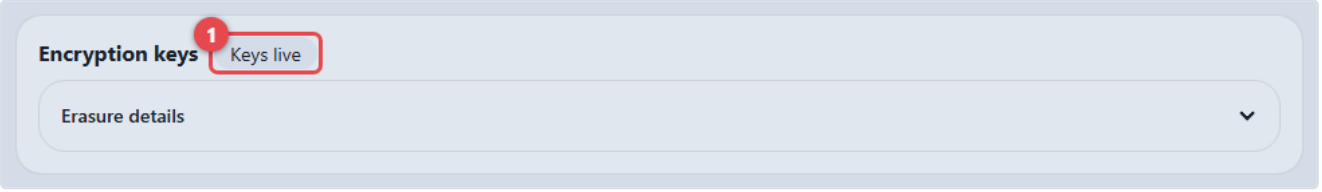
The erasure status of a contract

Each archive entry carries a badge in the **Encryption** column showing the state of its encryption:

- **Keys live:** the contract's content is retrievable.
- **Keys destroyed:** the keys have been destroyed; the content is permanently unreadable.

Clicking the badge unfolds the details below the row: for a destruction the time, triggering identity and justification, plus a table of the partner instances involved with **Peer**, **Status**, **Requested**, **Confirmed** and **Retries**.

The audit workspace shows the same state.

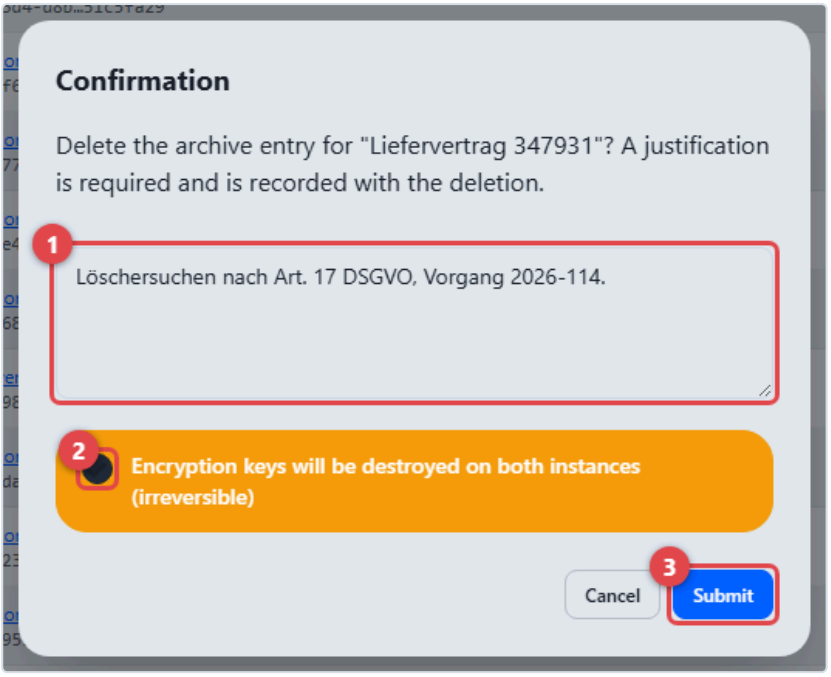


Once **Scope** stands at **Archive** and an identifier is entered, the **Encryption keys** section appears there with the same badge **(1)**. The unfoldable **Erasure details** area below it lists which partner organizations have confirmed the erasure.

Erasing contract content

An erasure request (for example under Art. 17 GDPR) is carried out by destroying the keys of the contract content. This is **final**: afterwards no one can decrypt the content any more, not even the operator.

In the archive entry's row choose **Delete**.



1. In the text field **(1)** enter the justification of the erasure request. The dialog names the affected contract at the top and points out that the justification is recorded with the erasure.
2. Tick the checkbox **(2)** next to the yellow notice "Encryption keys will be destroyed on both instances (irreversible)".
3. Only with justification **and** checkbox does **Submit (3)** become active; **Cancel** aborts without consequences.

After execution the view reports that the archive entry was deleted and the encryption keys destroyed; the entry disappears from the list, and the statistics bar is recalculated.

For a contract with a partner organization, the keys are destroyed on both sides. The audit workspace then shows the **Keys destroyed** state and, under **Erasure details**, the counterparty's confirmation.

What remains after erasure

- The contract stays visible in the contract overview with its master data (name, version, status). The content is erased, not the administrative record.
- An attempt to export the contract document ends with a clear message that the content was erased and the keys destroyed. The view stays usable.

- The erasure itself is a recorded event and verifiable in the audit workspace.

Running Audits (Auditor)

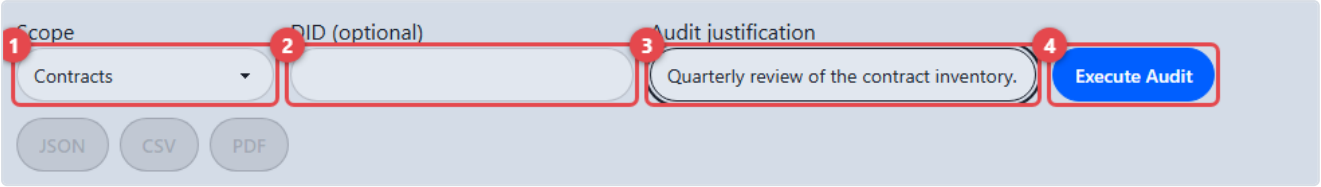
As an Auditor you run traceable audits across the data held by the DCS, at your choice across templates, contracts, signatures or archive operations. Every audit requires a justification and is itself recorded.

Prerequisites: You are signed in with the Auditor role. The side navigation shows **Audit**, **Archive** and **Compliance Viewer**. For an audit limited to a single object you need its identifier (DID).

An **Archive Manager** sees the same areas but can only audit the **Archive** scope in the audit workspace (see Managing the Archive and Retrieving Contracts).

The audit workspace

Open **Audit** in the side navigation.

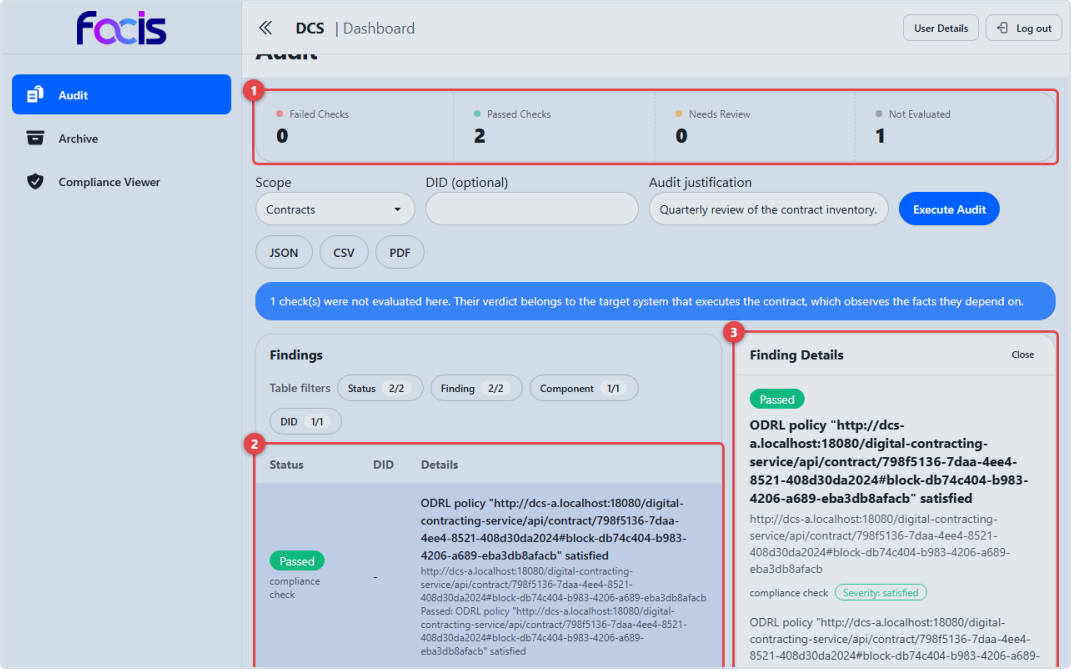


1. Under **Scope (1)** choose the audit scope: **Templates**, **Contracts**, **Signatures** or **Archive**.
2. Under **DID (optional) (2)** enter the identifier of a single object to limit the audit specifically to it. Without an entry, the whole stock of the chosen scope is audited.
3. Under **Audit justification (3)** give the justification for the audit; it is stored with the audit. Without a justification, **Execute Audit** stays locked.
4. **Execute Audit (4)** runs the audit. While it runs, the view reports "Executing audit...".

The audit is handed to the configured audit service exactly once. On a technical error there is no automatic retry, and no substitute findings are shown. Only start a new run once the cause is fixed.

For signature audits, only the details needed for the evidence are handed over (status, timestamps, checksums, references). The signature data itself does not leave the DCS.

Reading the audit result



An executed audit run over the **Contracts** scope. The counter bar **(1)** above the form summarizes the run by result kind: **Failed Checks**, **Passed Checks**, **Needs Review** and **Not Evaluated**. Below it, **Findings (2)** lists the individual findings; the row chosen here is highlighted and its audit evidence sits on the right under **Finding Details (3)**. The export buttons **JSON**, **CSV** and **PDF** are now unlocked because a run exists for this scope.

Above the result, a banner summarizes the run. If all checks passed, it reads "Audit passed. Every check returned a verdict, and none failed or needs review." If the run found nothing, the view reports "Audit completed. No matching entries were found."; a successful audit may therefore end without a finding. If there are unevaluated checks, a dedicated note points out that their verdict lies with the target system that executes the contract.

The **Findings** result list shows per row:

- **Status**, the verdict badge (**Passed**, **Failed**, **Needs Review**, **Not Evaluated**), below it the kind of finding, for example "compliance check".
- **DID**, the identifier of the audited object.
- **Details**, the audited statement in one sentence, below it the associated rule identifier and a short version of the result.

Use **Table filters** to narrow the table. Each of the four groups (**Status**, **Finding**, **Component**, **DID**) opens a list with checkmarks; **All** and **None** set all checkmarks of a group at once. The badge on the group shows how many of the available values are currently switched on. If no row matches any more, the table reads "No findings match the selected filters.". Before the first run it reads "Select a scope and execute an audit."

Clicking a row opens the associated audit evidence on the right under **Finding Details**: verdict, audited statement, severity and a list of the supporting details, for example **Checked**, **Rule ID**, **Requirement**, **Actual value**, **Expected value**, **Component** and **DID**. Details the audit service did not deliver are hidden

by the view instead of showing empty rows. At the very bottom, **Raw Details** unfolds the unaltered audit evidence. **Close** closes the detail column again; while no row is chosen, it reads "Select a row to inspect the corresponding audit evidence."

The audit history is kept in an immutable store. Immediately after a fresh change, a single entry may not be retrievable yet; run the audit again in that case.

Exporting an audit report

The **JSON**, **CSV** and **PDF** buttons export the audit report. They are locked until a successful run exists for the currently set audit scope (and, if entered, the object). If you change scope or identifier, they lock again.

1. First run a successful audit for the area you want.
2. Choose **JSON**, **CSV** or **PDF**.
3. Give the justification for the export and download the report.

The report is produced solely from the last stored audit run; the audit service is not called again on export. JSON contains the unaltered stored audit result, CSV and PDF present the same run in another form. The exported content is evidenced with its checksum in the audit trail. If the report cannot be stored verifiably, the application reports an error and does not release an unsecured report.

Checks at contract transitions

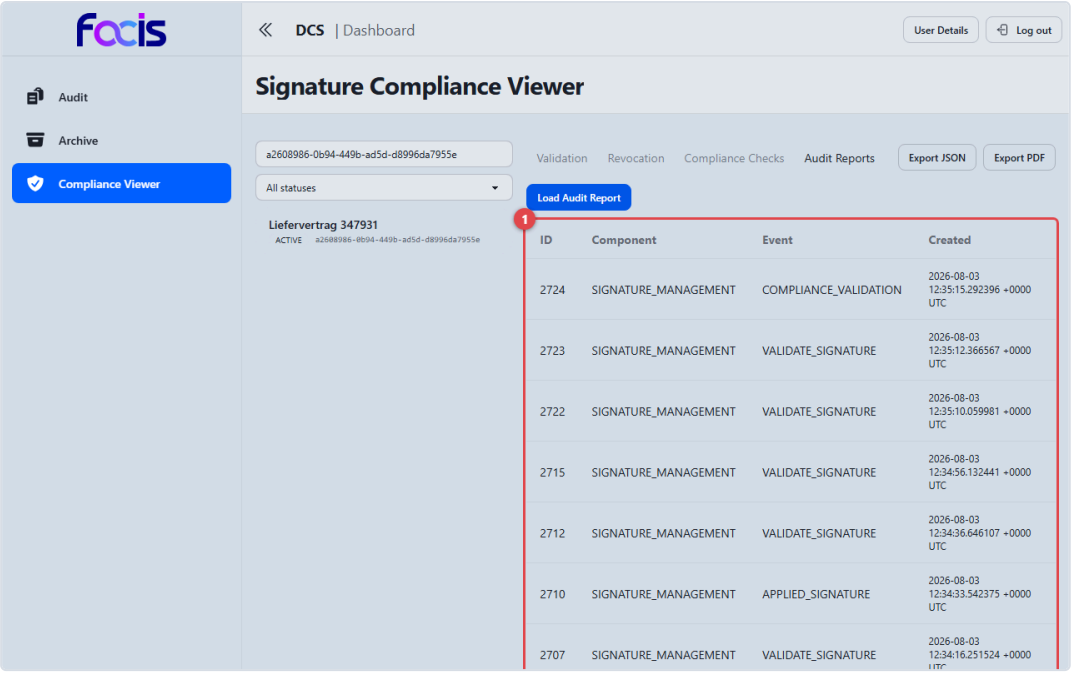
When a contract is submitted, offered, approved, signed or deployed, the rule state stored with the contract is checked automatically. No separate audit run is needed for this. Three outcomes are possible:

- **Passed:** the requested transition continues.
- **Manual review required:** the contract stays in its previous state. A Compliance Officer approves or discards the stored run with a justification; this decision happens outside the web interface (see Monitoring Compliance). After an approval, exactly the deferred transition continues.
- **Blocked:** the contract stays in its previous state. A new attempt only makes sense after the displayed rule violation or technical error has been fixed.

While a manual review is open, the contract must not be changed in content or state. Otherwise the continuation is rejected so that a decision is not applied to a contract that has changed in the meantime.

Signature audits in the Compliance Viewer

As an Auditor you additionally have access to the **Compliance Viewer** (see Managing Contracts). For you, the **Audit Reports** tab is the relevant one there.

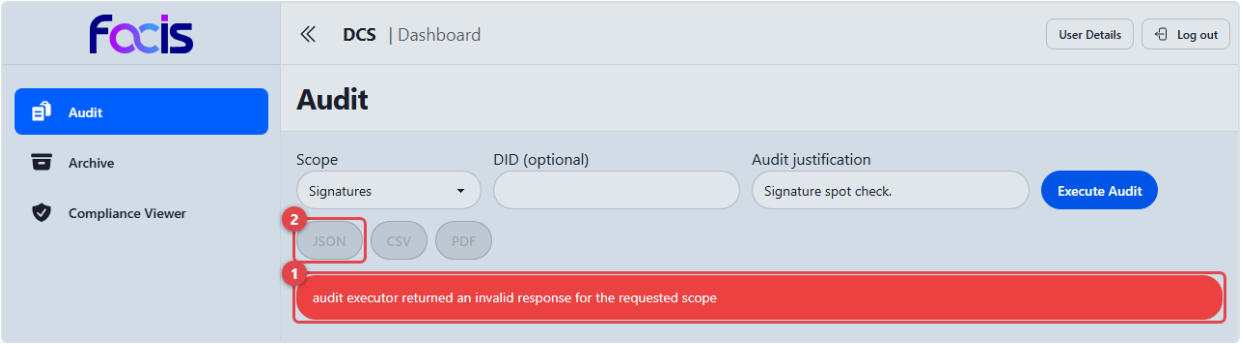


Load Audit Report loads the audit history of a contract's signatures as a table **(1)** with the columns **ID**, **Component**, **Event** and **Created**, newest entries first. Per entry you see the component involved, the event (in the example **APPLIED_SIGNATURE** for the applied signature and **VALIDATE_SIGNATURE** and **COMPLIANCE_VALIDATION** for the checks that followed) and the time. If nothing exists for the chosen contract, the table reads "No audit entries."

Load Audit Report requires the Auditor or Compliance Officer role; without it the button stays locked and a note next to it names the reason. The **Validate** and **Run Compliance** actions in the other tabs are reserved for the Contract Manager and not usable for you.

Visible error cases

- **Not authorized.** Sign in with the Auditor role. An Archive Manager may only use the archive scope in the audit workspace.
- **Invalid scope or missing justification.** Correct the selection or add a justification; the audit has not been handed over yet.
- **Audit service unreachable or timeout.** The run failed technically. The message appears as a red notice above the result area; the export buttons stay locked.
- **Invalid response from the audit service.** The run is rejected and not stored as a result.



This is what that looks like: the audit service responded, but its response did not match the audit request. The application shows the reason as a red notice, presents no result and keeps **JSON**, **CSV** and **PDF** locked. There is deliberately no substitute finding.

- **No matching stored audit run.** No successful run exists yet for scope and object. Run the audit first.
- **Rule state not resolvable.** A rule or profile version stored for this contract is missing or unreachable. The transition stays blocked; there is no fallback to another rule state.

Monitoring Compliance (Compliance Officer)

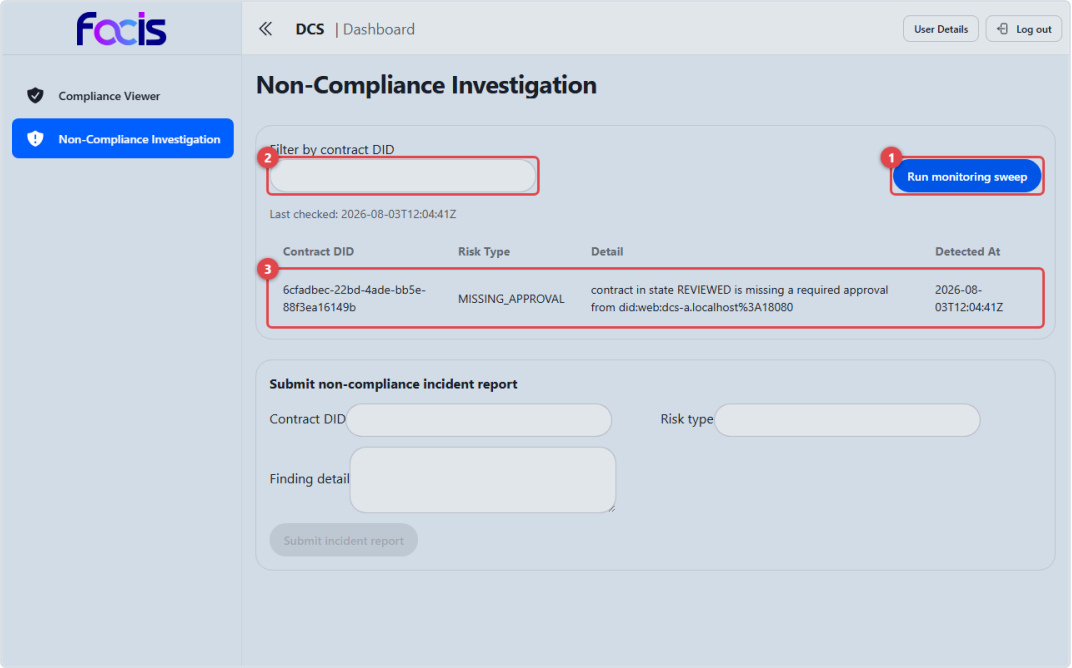
As a Compliance Officer you monitor the system for rule violations: you have the DCS sweep the stock for risks and report detected incidents as formal reports.

The **Non-Compliance Investigation** area is accessible to your role only; other roles are redirected automatically when they open it.

Prerequisites: You are signed in with the Compliance Officer role. The side navigation shows **Non-Compliance Investigation** and **Compliance Viewer**.

Running risk monitoring

Open **Non-Compliance Investigation** in the side navigation.

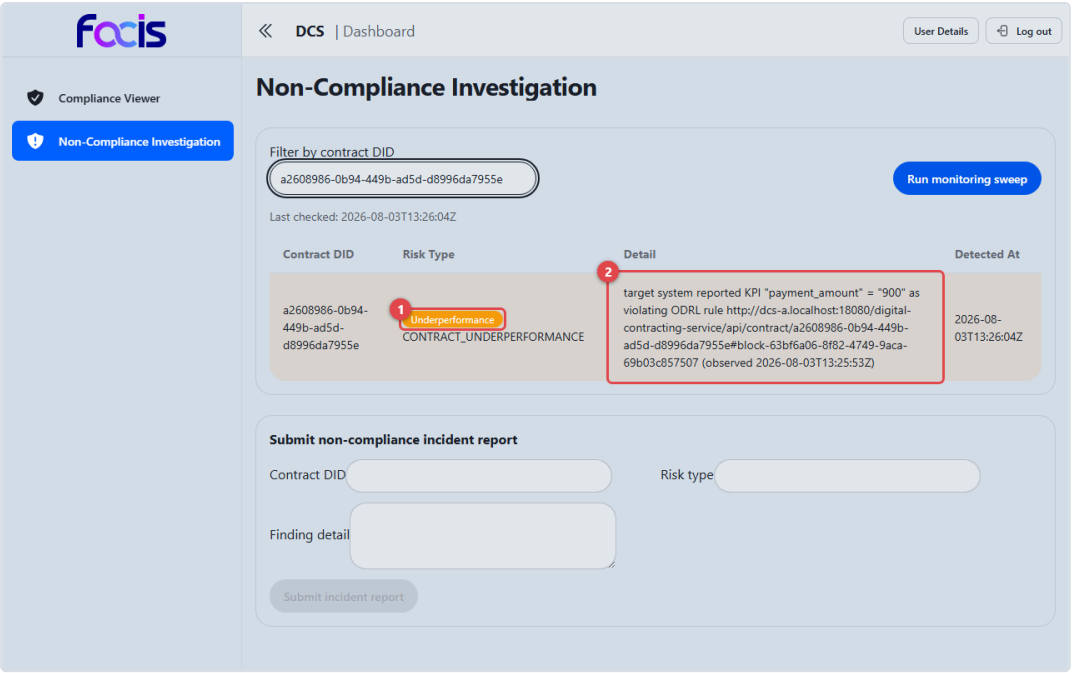


1. **Run monitoring sweep (1)** starts the monitoring run over the whole stock. While it runs, the view reports "Running monitoring sweep…"; afterwards the time of the last run is shown below it ("Last checked: …").
2. Via **Filter by contract DID (2)** you narrow the found risks to a specific contract.
3. The result table **(3)** shows per row **Contract DID**, **Risk Type**, **Detail** and **Detected At**. The figure shows a finding of the kind **MISSING_APPROVAL**; the description names the contract's state and the organization whose approval is outstanding.

The monitoring run detects, among others, the following risk kinds:

- **MISSING_APPROVAL:** a contract in review lacks a required approval.
- **CONTRACT_UNDERPERFORMANCE:** an activated contract misses a contractually agreed KPI. If the target system reports a measured value that violates the threshold stored machine-readably in the contract, the breach appears as an alert in the monitoring run and is additionally recorded in the

audit trail.



The row has a yellow background and carries the **Underperformance** badge **(1)** before the risk kind. The description **(2)** names the KPI, the reported value, the violated rule and the observation time, so the finding traces back to an individual contract clause.

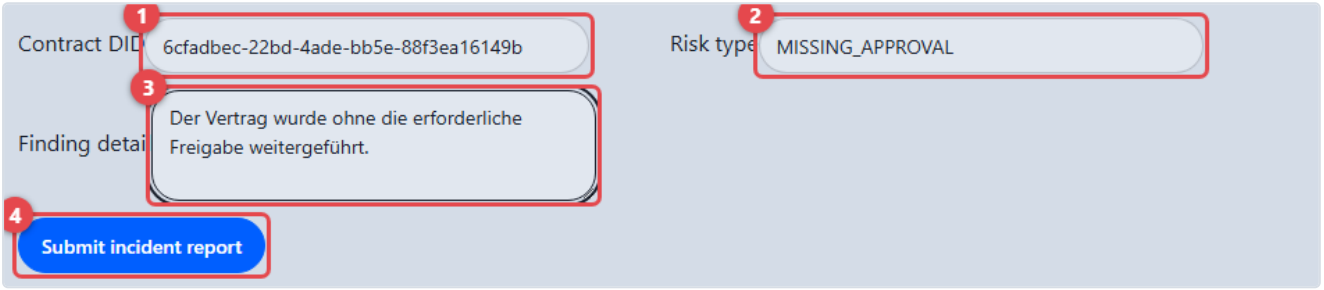
- **UNAUTHORIZED_ACCESS:** a party not involved in the contract tried to access it and was rejected. The rejection is recorded permanently; the monitoring run assigns the risk to the affected contract and names the rejected actor.

The view distinguishes clearly between the possible states and never claims "no risks" before it has checked:

- **Not run yet:** "Run a monitoring sweep to check for compliance risks."
- **Run without findings:** "Monitoring sweep completed. No compliance risks were found."
- **Filter without matches:** "No compliance risks match the current filter."
- **Technical error:** a red error message with the reason. An empty result list is not feigned in this case.

Reporting an incident

Below the monitoring you report detected violations in the **Submit non-compliance incident report** section as a formal incident report.



1. Under **Contract DID (1)** enter the identifier of the affected contract. The finding is firmly linked to this contract.
2. Under **Risk type (2)** state the risk kind (for example `MISSING_APPROVAL`).
3. Describe the finding under **Finding detail (3)** in your own words.
4. **Submit incident report (4)** submits the report; the button only becomes active once all fields are filled in.

After successful transmission the view confirms the report **(1)** ("Incident report submitted successfully."). The report is now permanently assigned to the contract and traceable for audits.

Deferred contract transitions

If the automatic check at a contract transition (submit, offer, approve, sign, deploy) results in "manual review required", the contract stays in its previous state until a Compliance Officer approves or discards the stored check run with a justification. After an approval, exactly the deferred transition continues; the audit service is not called again for it. A second approval does not start a parallel continuation.

While the review is open, the contract must not be changed in content or state. If it was changed anyway, the continuation is rejected because the earlier approval does not apply to the changed contract. The transition must then be requested anew for the current state.

There is no view for this decision in the web interface; it is made through the system interface of the DCS. Clarify the procedure with the operations team of your installation.

Signature checks

You additionally have access to the **Compliance Viewer** to inspect the signatures of signed contracts (see Managing Contracts). For you, the **Audit Reports** tab is usable there: **Load Audit Report** shows the audit history of a contract's signatures. The **Validate**, **Run Compliance** and **Revoke** actions are reserved for the Contract Manager; a note next to each names the missing role.

System Administration (Sys. Administrator)

As a Sys. Administrator you maintain three registries: the **target systems** that signed contracts are handed over to on activation, the **system users** under which connected external systems work on the DCS, and the **key inventory** that makes the instance's cryptographic keys visible.

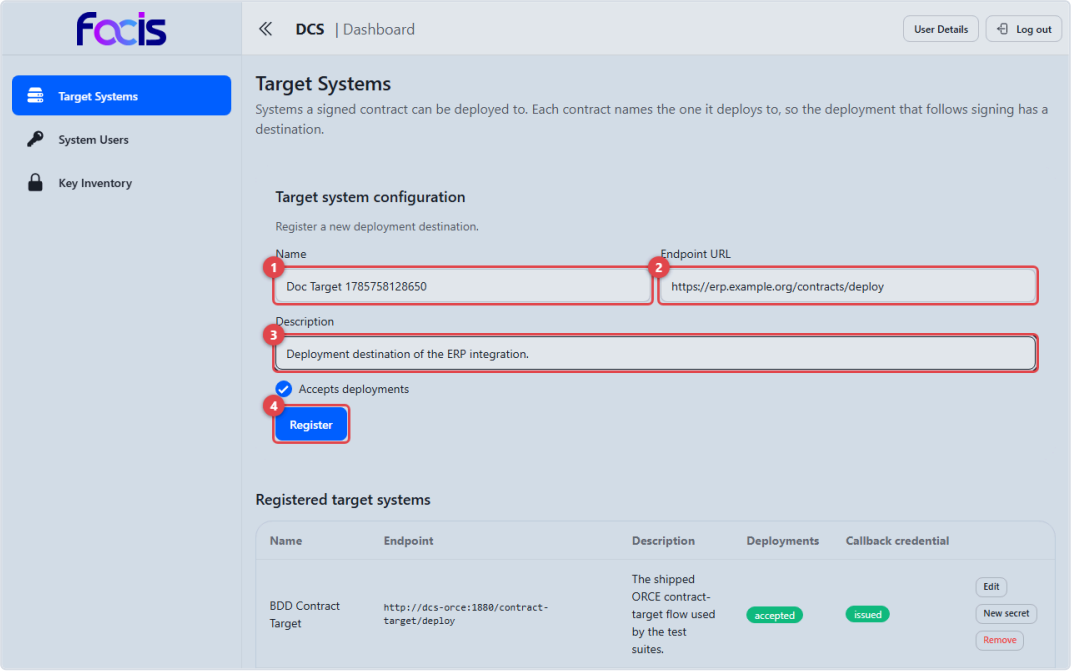
Prerequisites: You are signed in with the Sys. Administrator role. The side navigation shows **Target Systems**, **System Users** and **Key Inventory**. An **Integration Manager** sees **Target Systems** only.

Target systems (Target Systems)

Open **Target Systems** in the side navigation. The view consists of two parts: at the top the **Target system configuration** form, below it the **Registered target systems** table with the columns **Name**, **Endpoint**, **Description**, **Deployments** and **Callback credential**. The text below the form heading says what the form currently does: "Register a new deployment destination." when creating, "Change the selected deployment destination." when editing.

As long as no target system is registered, a note explains the starting position: a contract cannot be activated before a target system exists and the contract names it.

Registering a target system



1. Under **Name (1)** enter the display name under which the target system later appears in a contract's target selection.
2. Under **Endpoint URL (2)** enter the address deployments are transmitted to.
3. Optionally, **Description (3)** describes the entry in more detail.
4. **Register (4)** creates the target system; it appears in the table immediately.

The **Accepts deployments** switch controls whether the target system accepts deployments; the state appears in the **Deployments** column as **accepted** or **refused**.

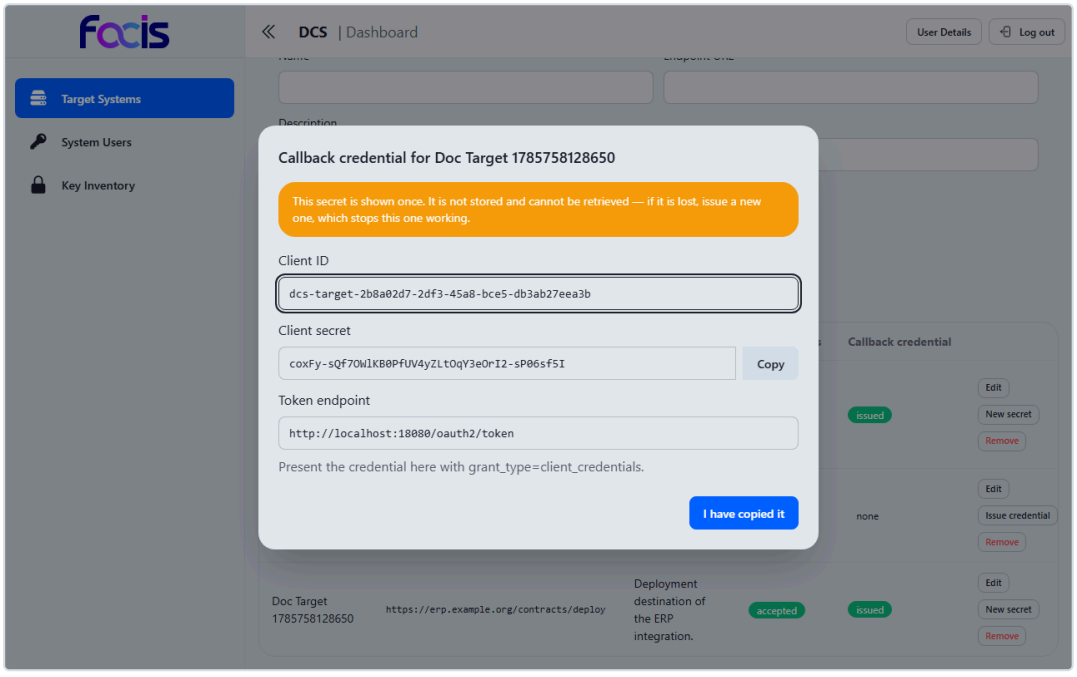
Name	Endpoint	Description	Deployments	Callback credential	
BDD Contract Target	`http://dcs-orce:1880/contract-target/deploy`	The shipped ORCE contract-target flow used by the test suites.	accepted	issued	Edit New secret Remove
Doc Target 1785757846980	`https://erp.example.org/contracts/deploy`	Deployment destination of the ERP integration.	accepted	none	Edit Issue credential Remove
Doc Target 1785758128650	`https://erp.example.org/contracts/deploy`	Deployment destination of the ERP integration.	accepted	1 none	Edit Issue credential Remove

In the **Callback credential** column, a new entry initially reads **none**: without credentials the target system cannot confirm a deployment.

Issuing credentials for confirmations

When a target system receives a deployment, it confirms receipt to the DCS and identifies itself with its own credentials. Each target system receives its own credentials so a confirmation unambiguously comes from the target system the deployment went to.

1. In the target system's row choose **Issue credential**.
2. A dialog shows the issued credentials, and it does so **once**.



The **Callback credential for …** dialog points out explicitly, in a yellow notice at the top, that the secret is shown only once: it is not stored and cannot be retrieved again. If it is lost, issue a new one; the old one then loses its validity. Below are **Client ID**, **Client secret** (with **Copy**) and **Token endpoint**, the address where the target system identifies itself with these credentials.

Hand the ID and the secret to the operators of the target system over a secure channel before closing the dialog with **I have copied it**. After that, the secret can no longer be seen anywhere in the application; the DCS does not store it.

3. After closing, the **Callback credential** column shows the note **issued**.

Via **New secret** you issue new credentials later, for example on suspicion of compromise. The previous secret loses its validity immediately; only the new one keeps working.

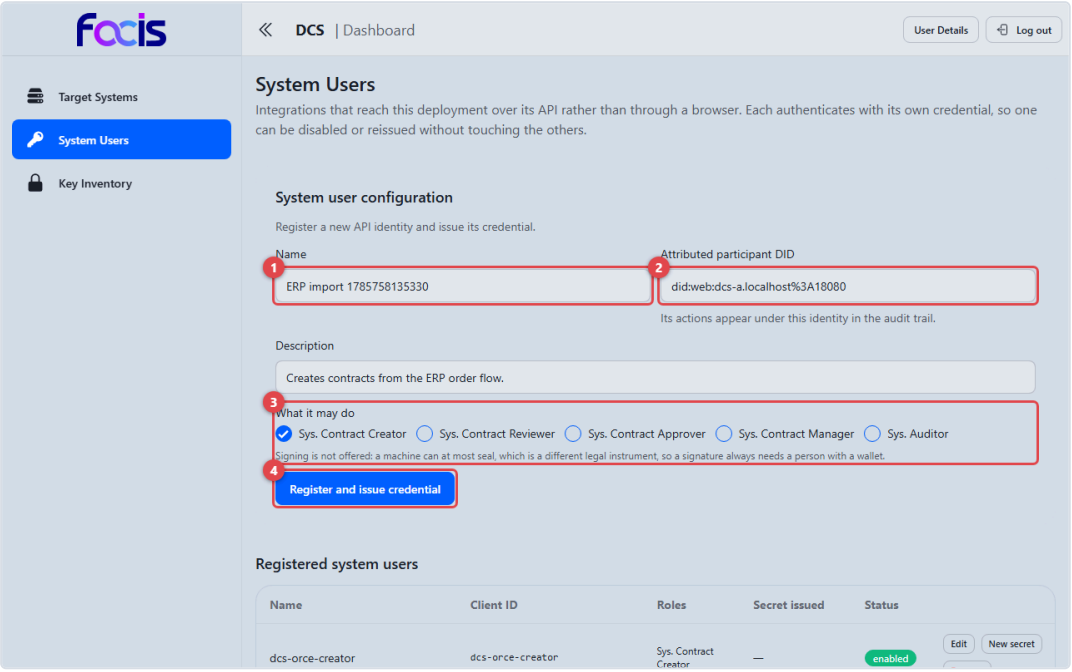
Editing and removing entries

- **Edit** loads an entry's values into the form; **Save changes** saves the change, **Cancel** discards it.
- **Remove** removes an entry from the registry. The confirmation dialog names the entry and points out that contracts still naming it must be repointed first.

If the system rejects an action, the corresponding message appears as a red notice above the form.

System users (System Users)

Open **System Users** in the side navigation. In the **System user configuration** form you register the technical users under which connected external systems work on the DCS, for example an upstream system that creates contracts automatically. Credentials are issued in the application and can be renewed at any time; no intervention in the installation is needed for that.



1. Under **Name (1)** enter a descriptive name for the connection.

2. Under **Attributed participant DID (2)** state the organization on whose behalf the system user acts. A note explains: its actions appear under this identity in the audit trail. The **Description** field below takes a more detailed description if needed.
3. Under **What it may do (3)** choose what the system user is allowed to do. Only the roles intended for automated accounts are offered: **Sys. Contract Creator**, **Sys. Contract Reviewer**, **Sys. Contract Approver**, **Sys. Contract Manager** and **Sys. Auditor**. A note explains why signing is not among them: a machine can at most seal; signing is always done by a person with a wallet.
4. **Register and issue credential (4)** creates the system user.

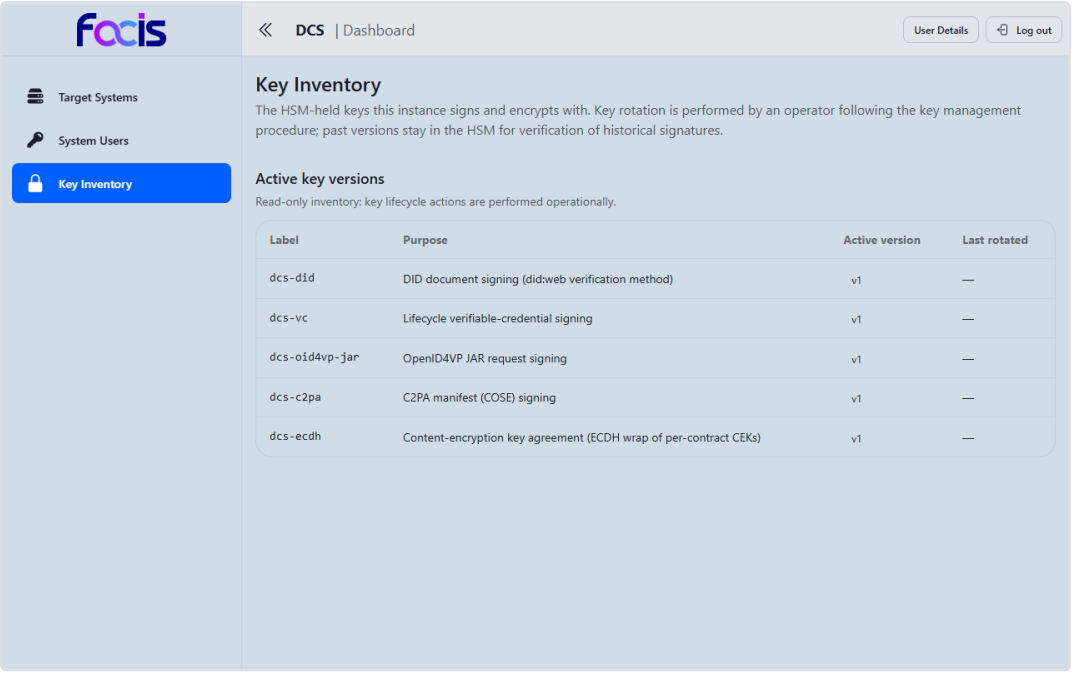
Immediately afterwards the same credential dialog appears as for the target systems: ID and secret are shown **once**. Note the secret before closing the dialog; after that it cannot be retrieved.

The **Registered system users** table below lists per entry **Name**, **Client ID**, **Roles**, **Secret issued** and **Status** (**enabled** or **disabled**). Further actions per row:

- **New secret** issues a new secret. The previous one loses its validity immediately; only the new one keeps working.
- **Edit** opens the entry for changes. Via the **May call this deployment** checkbox you disable a system user; the row then carries the note **disabled**, and its credentials are rejected from then on.
- **Remove** removes the entry. The confirmation dialog points out that the credentials thereby lose their validity and connections using them lose access.

Key inventory (Key Inventory)

Open **Key Inventory** in the side navigation.



The **Active key versions** table lists per key the label (**Label**), the purpose (**Purpose**) in plain language, the currently active version (**Active version**, for example **v1** as long as it was never rotated) and the time of the last rotation (**Last rotated**).

The view is read-only. The note above it says so explicitly: a key rotation is an operational procedure and is not triggered through the interface; earlier versions are kept so older signatures stay verifiable. This makes it possible to prove at any time which key material is in use in which version.

Troubleshooting and FAQ

This chapter collects the messages and edge cases you may come across day to day and says what you can do then.

Sign-in and access

The QR code on the sign-in page expires. QR code and sign-in link renew themselves as long as the tab stays open (roughly every five minutes). Do not close the page while you work in the wallet. If the code has expired anyway, reload the page.

I keep getting sent back to the home page. You opened a view for which you lack the role. **User Details** shows which roles you hold. Roles are not assigned in the application but through the credentials in your wallet.

"insufficient permissions: requires one of [...]". The server rejected a single action. Seeing a button does not by itself grant permission; every action is additionally checked on the server.

Sign-in does not complete despite presenting in the wallet. The presented credential is expired or revoked. The revocation status is checked at every sign-in; a revoked or expired credential is rejected. Have the issuing body issue you a valid credential.

Repeated failed attempts block access. After several rejected attempts with an invalid credential, the DCS rejects further attempts, even with a credential that is valid by then. Contact the operations team of your installation if the block persists.

"not authorized to access this contract". Your organization is not a party to this contract. Contracts are only visible to the parties involved. The rejected access is recorded permanently and appears in compliance monitoring as a risk of the kind UNAUTHORIZED_ACCESS.

Templates

"Global Name is required." / "Base Description is required." Mandatory entries are missing. The view jumps to the first incomplete field; nothing was saved. Add the entries and repeat **Create**.

I cannot find a "Verify" button in the review view. The formal verification is part of **Approve** and runs exactly once on every approval attempt. It cannot be skipped.

The verification dialog lists findings and offers no confirmation. This is intended: as long as verification finds something, the template cannot be forwarded. Have the listed items reworked and start the approval again.

"Verification could not be completed. Retry the verification before approving." Verification failed technically. **Retry** repeats it. The template was not forwarded.

My approved template does not appear when creating a contract. Approval (APPROVED) alone is not enough. The template must still be **registered** by the Template Manager (status REGISTERED / in use).

The "Data" tab says no shape libraries are registered. Semantic data objects only become available once a shape library has been published in the Semantic Hub. The Template Manager takes care of that.

A country is missing in a spatial constraint. The country and region list comes from the vocabulary and cannot be extended as free text. Contact the Template Manager, who maintains the Semantic Hub.

Contracts

"Offer to counterparty" is greyed out. The contract is not yet completely filled in. The tooltip names the missing mandatory fields. An offer must be complete before it leaves the house.

After offering, the button is gone. An offer is made exactly once (DRAFT → OFFERED). Further changes run through negotiation from then on.

"Submit" in the negotiation is locked. Change proposals are still open. The round can only be closed once all open proposals are decided. You cannot accept your own proposal; another authorized person decides it.

A received offer is not under Negotiation Tasks. That is the starting position: a negotiation task only comes into being when your side enters the round. Open the offer via **Contracts** → **View** and accept it via **Review offer** → **Accept offer**, or transmit a change; after that it appears under **Negotiation Tasks**.

"Accept offer" is rejected. An offer is accepted by the counterparty, not by the offering organization. On your own, self-offered contract the acceptance is rejected; the contract stays in the OFFERED status.

"Withdraw" is no longer offered. A contract can only be withdrawn up to approval (OFFERED, NEGOTIATION, SUBMITTED, REVIEWED). From APPROVED on, only termination by the Contract Manager remains (**Terminate**).

My saved negotiation draft is gone. Change Proposal consumes the private draft: what was proposed bindingly no longer exists as a draft. If you merely leave the view and come back later, the draft is still there.

The semantic pre-check reports items. Confirmation is not available then. Have the items fixed and start **Approve** again. If the dialog reports a technical error instead, **Retry verification** helps. If the pre-check succeeded and only the forwarding failed, **Retry submission** helps; your comment is kept.

The contract stays at SIGNED after signing. Most likely no target system is set. The **Deployment target** card points this out. The Contract Manager can add the target and trigger activation with **Deploy**.

"Deploy" is rejected. A contract is only deployed once all intended signatures are verifiably present and a target system is named.

A contract with three or more parties is not deployed. The proof of a counterparty's signature is kept per contract, not per signature field. With more than two parties the system therefore deliberately refuses execution instead of treating an incomplete signature situation as complete. Use contracts with two parties.

"insufficient permissions" flashes up when creating a contract. The **New Contract** view tries to load the chosen template as a preview; reading a template is reserved for the template roles. The message disappears by itself, and the preview area stays empty. Creating the contract is not affected: choose the template and continue with **Create**.

"Terminate" cannot be completed. The confirmation dialog requires a reason (**Reason**). Without a reason nothing is carried out.

The complete package ("Export bundle") is rejected. The package is only produced complete. If a part is missing, for example because no contract document has been generated yet for one of the contracts, the red message names the reason; an incomplete package is not produced.

Signing

The contract is not in my signing list. Only approved contracts (APPROVED) appear there. Ask the Contract Approver.

"Waiting for the counterparty to settle this version ...". For a contract with a partner organization, your signature is only allowed once the counterparty has bindingly confirmed the same contract state by closing its own negotiation round. This is not a rejection of you personally: wait until the counterparty's confirmation arrives, then signing opens by itself. If renegotiation happened, the counterparty must confirm the new version again.

Step 3 or 4 is locked. The steps are deliberately unlocked one after another. Only the successful integrity check (**Verify**) unlocks the download, and only the download unlocks the upload.

"... does not identify the verified signatory". The certificate the PDF was signed with names someone other than the identity credential presented in the wallet. Use a signing tool whose certificate is issued in your own name and upload again. The contract stays unsigned.

My power of attorney is rejected. The power of attorney must name you and the signing organization, be valid (not revoked or suspended), contain a checkable status proof and trace back to a trust anchor accepted for your environment. If any of these points is missing, the signing process does not continue. Contact the issuing body.

The signing process aborts with an expired deadline. A deadline applies to every signing process. If the wallet confirms only afterwards, the presentation is rejected. Restart via **Open & sign**.

My PDF program reports "changed after signing". This is expected and not an error. After signing, the DCS attaches a provenance proof to the document so the evidence chain stays complete. Checking programs treat every addition after the signature as a change. The signature itself remains valid and verifiable unchanged.

Audit, compliance and archive

The export buttons (JSON/CSV/PDF) are grey. An export requires a successful audit run for exactly the configured audit scope. If you change scope or identifier, the buttons lock again until another audit has run.

The audit run fails technically. There is deliberately no automatic second attempt and no substitute finding. Only start a new run once the cause is fixed.

A record changed just now does not show up in the audit. The audit history is kept in an immutable store; immediately after a change a single entry may not be retrievable yet. Repeat the audit.

A contract transition hangs ("manual review required"). A Compliance Officer must decide the stored check run; this decision happens outside the web interface. Do not change the contract in the meantime. Otherwise the continuation is rejected, and the transition must be requested anew for the current state.

"Content erased" on export. An erasure request was carried out for this contract, and the encryption keys were destroyed; the content is permanently unreadable. The contract's master data stays visible. Recovery is impossible.

The delete button in the archive stays inactive. The dialog requires both: a justification and the explicit checkbox on irreversibility.

The annotation of an archive entry is not saved. After the first dialog (short description), a second one for the tags must follow. If it stays away, the process aborts, and the **Annotation** column keeps showing "Not annotated". Contact the operations team of your installation; the annotation can be added there through the system interface.

The signature validation reports INDETERMINATE instead of passed. The EU validation procedure reaches no final verdict, usually because it lacks the signatory's certificate chain (**NO_CERTIFICATE_CHAIN_FOUND**). The integrity findings below still show whether document and signature are intact. Whether the trust chain is fully installed in your installation is a question for the operations team.

The archive dashboard does not yet show a contract signed just now. The archive entry comes into being shortly after signing. Reload the view after a moment.

Further known limitations

Check of the displayed contract text. During signing, the system checks whether the visible text matches the machine-readable content. This check can be circumvented with a deliberately crafted PDF document. When in doubt, additionally check the contract content in the **Machine-readable view**; it is generated directly from the stored contract data.