SQL Change Guard User Guide

Version: 1.0

Scope: Installation, configuration, daily use, audit

Audience: Anyone using the product for the first time

Section 0. Introduction and Terms

0.1 What SQL Change Guard is

SQL Change Guard is a governance application that records every intervention made on your databases, ties it to an approval flow and produces the evidence for it.

The typical situation in an organisation is this: a developer writes a SQL script, emails it to a manager, the manager replies "go ahead", and a database administrator runs the script on the production server. Three things are missing from this flow:

1. Nobody systematically examined what the script does.

2. There is no record of who approved it, when, and under which rule.

3. Six months later, when an auditor asks "who approved this change", the answer is searched for in mailboxes.

SQL Change Guard closes these three gaps. The script enters the application, is parsed automatically, its risk is calculated, the required approval steps are created from the rule set, it is executed by the application after approval, and every step is written to an immutable audit trail.

0.2 Three focus areas

The product works under three headings. Throughout the guide it is stated which section serves which heading.

Focus area	**Meaning**	**Sections in this guide**
Database Change Governance	Managing work that changes the database structure. Who changed what, who approved it, when it ran.	10, 11, 12, 14, 15, 16
Production Query Governance	Managing work that reads data from live environments. Who ran which query, who received the result.	13
Data Governance & Compliance	Protecting personal and sensitive data, producing audit evidence.	9, 13, 17, 18, 20

The overall positioning of the product is Database Operations Governance.

0.3 Who uses it

Role	**Priority sections**
Developer	11, 12, 14, 22, 26
Database administrator	6, 11, 12, 15, 25, 26

Role	Priority sections
Change manager	10, 11, 19, 20
IT manager	2, 3, 7, 8, 25
Auditor	17, 18, 20, 23
System manager	2, 3, 4, 5, 6, 7, 8, 9, 25

0.4 Glossary

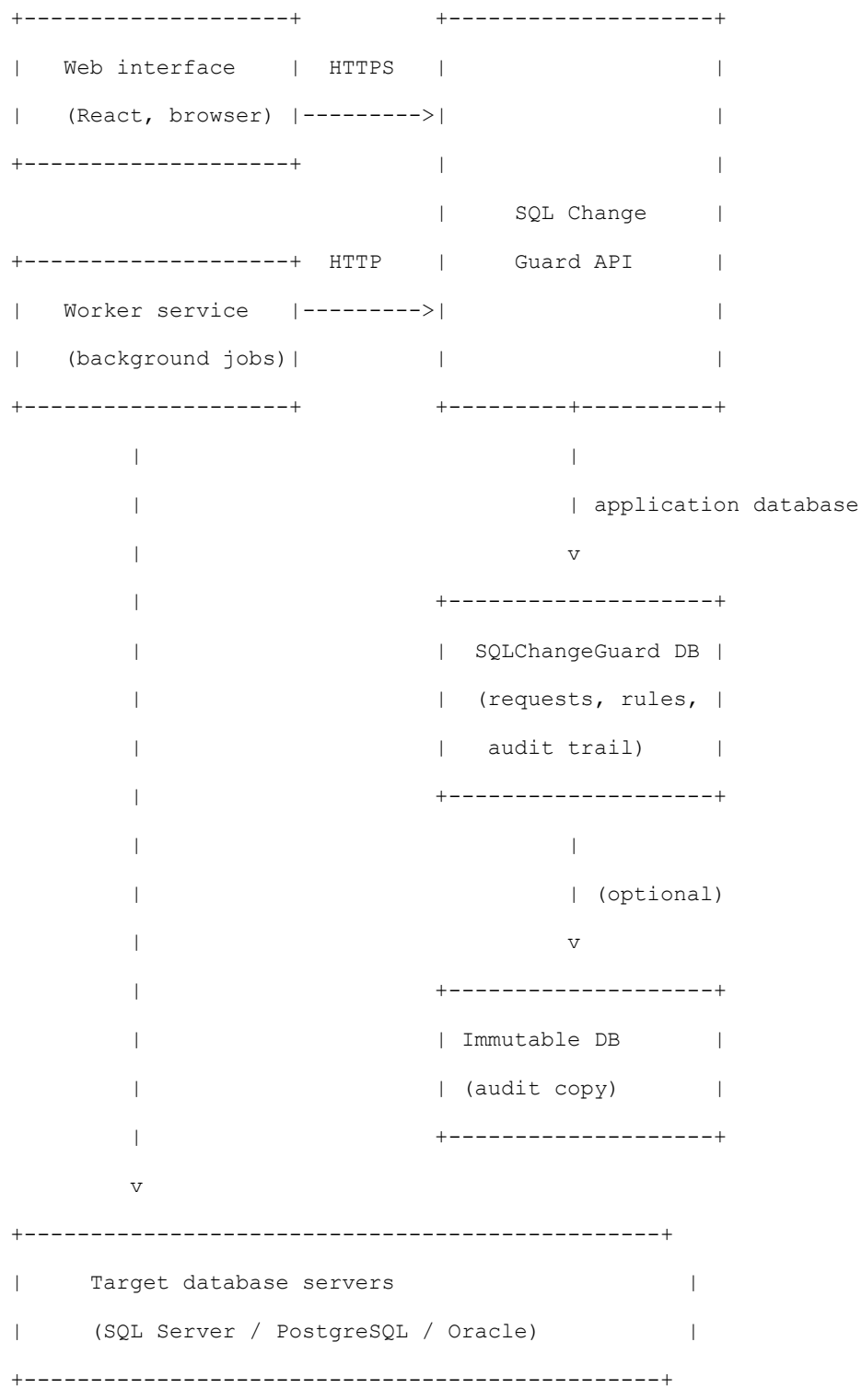
Term	Description
Change Request	A record opened so that one or more SQL scripts run on one or more target servers. Abbreviated CR.
Query Result	A request type opened to read data from a live environment. The result is delivered as an encrypted file. Abbreviated QR.
Rollback request	A request opened against a change request in order to reverse it.
Script	A single SQL text inside a request. A request may carry several scripts.
Parsing	Analysing a script as structure rather than as text. Which operation touches which object comes from here.
SQL standard	A rule searched for inside scripts. Example: "is there an xp_cmdshell call".
Critical object	A database object the organisation has explicitly flagged. Requests touching it behave differently.
Risk band	The risk level of a request. Five steps: Info, Low, Medium, High, Critical.
Risk score	The readable 0-100 projection of the band. A higher number means safer.
Approval policy	The rule set defining how many people, holding which role, must approve in which situation.
Approval step	A single approval request produced by a policy.
Four eyes	The principle that an action is completed by at least two different people.
Segregation of duties (SoD)	The principle that the requester, the approver and the executor are different people.

Term	Description
Emergency request	A request type where the normal approval path is shortened and a written reason is mandatory.
Sandbox	Trying a script on a test server before it goes live.
Release Package	Running several requests together in a defined order.
Audit log	The immutable record of every event in the system.
Evidence Dossier	A ZIP package containing the full lifecycle evidence of a single request.
Masking	Making sensitive values in a query result unreadable.
Worker	The Windows service that runs scheduled work in the background.

Section 1. Architecture and Components

1.1 Overview

The system consists of four main parts. They can be installed independently.



1.2 Components and their duties

Component	What it does	What happens if it stops
API	All business rules run here. Authorisation, approval flow, risk calculation, audit writing.	No screen works.
Web interface	The screens the user sees. Contains no business rules, asks the API for everything.	Users cannot reach the screens, but background work continues.
Worker service	Scheduled execution, server connection checks, notification delivery, audit integrity checks, maintenance jobs.	Scheduled requests do not run, email is not sent, connection status is not refreshed. Manual execution still works.
Application database	Requests, rules, users, audit trail.	The system stops completely.
Immutable database	A second copy of the audit trail written to a separate database. Optional.	The audit trail still exists in the main database, but the second copy comparison cannot be made.
Target servers	The databases where changes and queries actually run.	Only requests for that server fail.

1.3 Supported database types

DatabaseType values:

Value	Type
1	SQL Server
2	Oracle
3	PostgreSQL

These are used in two separate places and must not be confused:

1. The application's own database: selected with the DatabaseProvider setting. The installation described in this guide runs on SQL Server.

2. Target servers: each server definition has its own type. One installation can manage SQL Server, Oracle and PostgreSQL servers at the same time.

Which target types are available comes from the license. See Section 3.

1.4 Worker jobs

The worker service runs on the Quartz scheduler. Each job has its own period, read from the Workers section of the API appsettings.json.

Job	What it does	Setting key	Default
MainWorkJob	Four things in order: (1) runs due release packages, (2) runs queued requests, (3) runs pending sandbox jobs, (4) runs due scheduled requests.	Workers:MainWorkSeconds	30 seconds
ConnectionCheckJob	Tries to connect to defined servers and updates status.	Workers:ConnectionCheckSeconds	60 seconds
NotificationJob	Sends queued emails and notifications.	Workers:NotificationFlushSeconds	60 seconds
MaintenanceJob	Runs maintenance tasks (MaintenanceTasks table) on schedule.	Workers:MaintenanceSeconds	30 seconds
AuditIntegrityJob	Verifies the audit chain and raises a warning on corruption.	Fixed period in code	-
HeartbeatJob	Reports to the API that the worker is alive.	Fixed period in code	-

Enable and disable flags:

Flag	Effect
Workers:MainWorkEnabled	If false, automatic execution stops entirely.
Workers:ConnectionCheckEnabled	If false, server connection status is not updated.
Workers:MaintenanceEnabled	If false, maintenance tasks do not run.

MainWorkJob never runs twice at the same time. A new cycle does not start before the previous one finishes. This is the first layer preventing a request from being executed twice. The second layer is at the database level: before starting execution the worker claims the request, and another worker cannot claim the same one.

1.5 Scaling and deployment

Component	How many instances	What to watch
API	Several instances can run. Session information travels inside the token and no session state is kept on the server.	Rate limiting counters are per instance. With two instances the effective per-user limit doubles. Adding instances does not increase save speed; the reason is explained in section 1.6.
Web interface	Several instances can run.	The address of every instance must be listed in AllowedOrigins.
Worker	Exactly one instance must run.	Claiming a request is atomic at the database level, so two workers can never take the same request at the same time. There is however no lease mechanism between workers: recovery of interrupted executions relies on a time threshold alone. See Section 25.7.
Application database	A single database.	High availability is solved at the database server level (a failover cluster for example).

Practical capacity notes:

- The runtime of a request depends on the script on the target server; the application-side upper bound is the MaxQueryTimeout parameter.

- Because worker cycles never overlap, a very long request delays the other work in the same cycle. Schedule long maintenance work outside busy hours.

- Report queries have their own timeout (Reports:QueryTimeoutSeconds) and run against the application database. Running reports over very wide date ranges puts load on that database.

1.6 Measured capacity and sizing

The values below are not estimates; they were measured by putting real requests through the end to end flow, and the measurement was repeated five times. How to read the table and the recommended server are covered in the heading right below it.

Measure	Value
Request save time, median	0.1 to 0.6 seconds
Request save time, p95	0.2 to 0.7 seconds
Requests saved per second, single user	2 to 9

Measure	Value
Database growth per request	about 57 KB

The range depends on what else the measurement machine is doing at the time. On an idle machine a save takes a tenth of a second, on a busy machine about half a second. From the user's point of view both are instant.

How to read this table, and the recommended server

The published values are measured values and they are a lower bound. The measurement was made on a four core laptop with 8 GB of memory, with the application and the database on the same machine. On a dedicated server the values are not expected to be worse. No separate measurement was made on a server, so this document publishes no estimated server figure: every number written here was actually measured.

Recommended minimum installation:

Component	Minimum	Note
Application server (API and Worker)	4 cores, 8 GB memory, 20 GB disk	The web interface can run on the same server.
Application database server	4 cores, 16 GB memory, SSD	Around 10 GB of data space is enough for five years of retention.
Network	The application server and the database server must be on the same local network	The reason follows below.

The network distance between the application and the database matters more than the choice of processor. A single request save makes around a hundred database round trips. If the latency between them is 0.3 milliseconds, that adds up to thirty milliseconds and goes unnoticed. If the same latency is 5 milliseconds (a different data centre or behind a VPN), half a second is added to the save time for that reason alone. Keep the database server close to the application server in network terms.

Storage planning

At 57 KB per request:

Requests per day	Growth per year
20	about 400 MB
50	about 1 GB

Requests per day	Growth per year
100	about 2 GB

This is raw growth. Once the maintenance job archives old records, growth slows down.

The number of rows written per request is higher than it first appears, because every change request also produces its own rollback request and the database audit trail stores each write as a separate row.

Concurrent use and team size

To guarantee that records are chained to one another, the audit trail chain serialises save operations. As the number of users saving at the same moment grows, the save takes longer.

Measured behaviour, totals over five runs:

People saving at exactly the same moment	Attempted	Errors returned to the user
1	40	0
2	80	0
4	160	3
8	320	4

Up to two people there are no errors at all. When four or eight people save at exactly the same moment, one or two requests in a hundred return an error to the user. Pressing the save button again is enough; no data is lost and no half record is created.

The measured system wide save ceiling is about 20 requests per second. This ceiling comes from the audit trail chain, not from the processor. One practical consequence follows: running more than one copy of the API does not increase save speed, because the serialisation point is in the database and not in the application. Extra copies spread the load on the read side.

How large a team it supports. Team size is not the practical limit. A team of several hundred people is supported. What sets the limit is not headcount but how many changes the team actually writes. Twenty saves per second is hundreds of thousands of saves in an eight hour working day, and enterprise database change volume is nowhere near that. Four people pressing save in the very same second only becomes routine once the daily request count reaches the thousands. For a team producing fifty requests a day, saving at the same moment happens on the order of once a week. Spreading requests through the day reduces the chance further during busy release hours.

Number of servers

The number of managed servers does not affect save speed. The limit comes from the connection health check: every sixty seconds the worker probes all defined servers one after another, and waits up to thirty seconds for a server it cannot reach.

Environment	Recommended number of servers
Servers on the local network	100 and above
Remote or cloud servers	30 to 40

Unreachable servers lengthen the cycle. Setting decommissioned servers to inactive keeps the health check finishing on time.

Summary of limits

Limit	Value	What sets it
Team size	several hundred people	Not headcount, but the number of requests per day
People saving at the same moment	no errors up to 2; at 4 and 8 people, 1 to 2 retries per hundred requests	The audit trail chain
System wide save ceiling	about 20 requests per second	The audit trail chain
Requests per day	at 5,000 requests the system is busy 2 percent of the time	The ceiling above
Managed servers	100 and above on the local network, 30 to 40 remote or cloud	The connection health check cycle
Storage	about 57 KB per request	Measured

These values were measured for the request save path, on a newly installed database. As data accumulates over time, list and report queries become dependent on data volume; archiving through maintenance tasks is recommended for that reason.

Section 2. Installation

This section covers a from-scratch installation. The order matters.

2.1 Prerequisites

Requirement	Description
Application server hardware	At least 4 cores, 8 GB memory, 20 GB disk. The API, the Worker and the web interface can run on the same server.
Database server hardware	At least 4 cores, 16 GB memory, SSD. Around 10 GB of data space is enough for five years of retention.
Network proximity	The application server and the application database server must be on the same local network. The reason is explained in section 1.6.
.NET 10 runtime	The API and the Worker are built against this version (net10.0).
SQL Server 2016 or later	For the application database. No feature requiring a newer version is used.
Windows server	The worker runs as a Windows service. Evidence dossier PDF generation also uses Windows fonts.
Network access	The API server must reach the target database servers. Default ports: 1433 for SQL Server, 1521 for Oracle, 5432 for PostgreSQL.
SMTP server	For notification email. Not mandatory, can be switched off.
Service account	The account the worker runs under. It must be able to write to the application database and to the QR file folder.

2.2 Installation steps

```
1. Create the database
       |
       v
2. Edit the API appsettings.json
       |
       v
3. Start the API  (schema and default data are created automatically)
       |
       v
```

```
 4. Upload the license file
        |
        v
 5. Sign in as the first administrator, change the password
        |
        v
 6. Define the servers
        |
        v
 7. Add the users
        |
        v
 8. Set the parameters and approval policies
        |
        v
 9. Start the worker service
        |
        v
10. Run the verification checklist
```

2.3 Creating the database

Creating an empty database is enough. Tables, stored procedures and default data are installed automatically when the API starts for the first time.

The setting that enables this:

```
"Database": {
  "AutoMigrate": true
}
```

When AutoMigrate is true, the API applies the following files in order at startup:

File	Content
001_Tables.sql	All tables and indexes
002_Migrations.sql	Schema version tracking
003_SeedData.sql	Roles, default administrator, parameters, SQL standards, report definitions, maintenance tasks

File	Content
004_PostSeedFixes.sql	Default data corrections
005_StoredProcedures.sql	Report stored procedures
006_TableWriteLogs.sql	Database level write log and its triggers

The scripts are written to be re-runnable. Existing records are not inserted again and your own changes are not overwritten.

If AutoMigrate is set to false, a database administrator must apply these files manually. Organisations with strict change control usually prefer this.

2.4 appsettings.json field by field

The table below describes every section of the API appsettings.json. In production, every value reading REPLACE_IN_PROD must be changed.

2.4.1 ConnectionStrings

Key	What it does	Example
ConnectionStrings:Default	Connection string for the application's own record database.	Server=SQLSRV01;Database=SQLChangeGuard;User Id=scg;Password=...;TrustServerCertificate=True
DatabaseProvider	Provider of the application database.	SqlServer

The application's own record database runs on SQL Server. PostgreSQL and Oracle are managed target databases; their connection details are not kept here but in the records on the Servers screen.

If wrong: the API cannot connect at startup and writes a connection error to the log file. No screen opens.

2.4.2 ImmutableDb

The separate database where the second, immutable copy of the audit trail is written.

Key	What it does
ImmutableDb:Enabled	When true, every audit record is also written to the second database.
ImmutableDb:ConnectionStrings:Default	Connection string of the second database.

It is recommended that the user of this database is granted INSERT permission only. The purpose is that someone with full access to the main database cannot retroactively correct the audit trail. See Section 17.

2.4.3 Auth

Key	What it does	Default	Example effect
Auth:Mode	Authentication mode. Local means local users; fill in the LDAP section to use a directory.	Local	
Auth:AccountLockoutMaxAttempts	How many wrong password attempts lock the account.	5	With 5, the sixth wrong attempt locks the account.
Auth:AccountLockoutMinutes	Lock duration in minutes.	15	With 15, a locked account unlocks itself after 15 minutes.
Auth:Password:MinLength	Minimum password length.	6	
Auth:Password:RequireUppercase	Is an uppercase letter required.	false	
Auth:Password:RequireDigit	Is a digit required.	true	With true, "secret" is rejected and "secret1" is accepted.
Auth:Password:HistoryCount	How many previous passwords cannot be reused. 0 = no check.	0	With 3, a user cannot pick any of their last three passwords.
Auth:Password:ExpiryDays	How often the password must change. 0 = never expires.	0	With 90, the user is forced to change the password after 90 days.
Auth:Password:ResetTokenExpiryMinutes	Validity of the password reset link.	30	
Auth:Password:ResetBaseUrl	Root address used in the reset email link. Write the address of the web interface.	http://localhost:5173	If wrong, the user lands on an invalid page when clicking the link.

2.4.4 Jwt

Settings for the access token issued when a user signs in.

Key	What it does	Default
Jwt:Issuer	Name of the token issuer.	SCG
Jwt:Audience	Name of the token consumer.	SCG-Clients
Jwt:AccessTokenMinutes	Lifetime of the access token in minutes. When it expires it is renewed with the refresh token.	120
Jwt:RefreshTokenHours	Lifetime of the refresh token in hours. When this also expires the user signs in again.	48
Jwt:SigningKey	Signing key of the token. Must be changed in production.	REPLACE_IN_PROD

Why SigningKey matters: the token is signed with this key. Anyone who knows it can produce a valid token for any user, that is, enter the system without knowing a password. Use a random value of at least 32 characters.

If the key is changed: every open session becomes invalid and everyone signs in again. No data is lost.

2.4.5 Security (encryption keys)

Key	What it does	If lost
Security:ServerPasswordKey	AES-256 key used to encrypt database passwords in server definitions at rest. A 32-byte value, base64 encoded.	Stored server passwords cannot be decrypted. Each password must be re-entered.
Security:QrZipPasswordKey	AES-256 key used to encrypt query result ZIP passwords.	Passwords of old result files cannot be displayed.
Audit:ActiveKeyId and Audit:HmacKeys	The HMAC key ring used to sign audit records.	Old records cannot be verified and are marked "not verified".

How to generate these keys: encode a random 32-byte value as base64. With PowerShell:

```
$bytes = New-Object byte[] 32
[System.Security.Cryptography.RandomNumberGenerator]::Create().GetBytes($bytes)
[Convert]::ToBase64String($bytes)
```

It is recommended to supply these keys through environment variables or user secrets rather than leaving them in plain text inside appsettings.json.

Key rotation trap: if ServerPasswordKey changes, previously encrypted values cannot be decrypted with the new key. Plan for re-entering every server password before changing the key. The audit HMAC key is different: the key ring can hold several keys, and old records continue to be verified with their own key id.

2.4.6 License

Key	What it does	Default
License:LicensePath	Path of the license file.	license.scglicense
License:PublicKeyPem	The public key that verifies the license signature. Not to be changed.	Ships with the product
License:ExpiryWarningDays	How many days before expiry a warning is raised.	30

With ExpiryWarningDays set to 30, once 29 days remain the license status becomes "Expiring" and a warning appears in the interface. The system keeps working.

2.4.7 Mail

Key	What it does	Note
Mail:Enabled	Is email delivery on.	If false, no email is sent, but the record is still kept.
Mail:SmtpHost	SMTP server address.	
Mail:SmtpPort	SMTP port.	587 is common.
Mail:Username, Mail:Password	SMTP credentials.	
Mail:FromAddress, Mail:FromName	Sender address and display name.	
Mail:EnableSsl	Use an encrypted connection.	Should be true in production.
Mail:DevMode	When true, all email goes to a single test address instead of the real recipients.	Must be false in production.
Mail:DevOverrideEmail	The test address used while DevMode is on.	

The most common production mistake is leaving DevMode on. In that case requesters never receive email; everything lands in the test mailbox.

2.4.8 Ldap

Used for signing in through the corporate directory.

Key	What it does
Ldap:Host	Directory server address.
Ldap:Port	Port. 389 unencrypted, 636 encrypted.
Ldap:UseSsl	Use an encrypted connection.
Ldap:BaseDn	Root where the search begins.
Ldap:BindDn, Ldap:BindPassword	The service account that searches the directory.
Ldap:Domain	Short domain name.
Ldap:TimeoutSeconds	Connection timeout.
Ldap:AllowUntrustedCertificate	Allow untrusted certificates. Should be false in production.

The difference between LDAP users and local users is described in Section 4.8.

2.4.9 Other settings

Key	What it does	Default
Display:TimeZone	The time zone used for times in documents and other server-side output.	Europe/Istanbul
AllowedOrigins	Addresses where the web interface runs. Requests from addresses not listed here are rejected.	localhost addresses
Mfa:Enabled	The master switch for two-step verification. When off, no user can set up two-step verification and no code is asked for at sign-in. Being on does not by itself make a code mandatory; see below.	true
Mfa:Method	Method. Totp means an authenticator application.	Totp
Mfa:Issuer	The name shown in the authenticator app.	SQLChangeGuard

How two-step verification takes effect: even with the master switch on, a code is asked only from users who have set up two-step verification on their own account. Setup is per user and is performed by the user themselves. Turning the master switch off also disables existing setups.

Two-step verification cannot be set up for directory (LDAP) accounts; authentication for those accounts belongs to the corporate directory.

QueryResult:OutputPath	**Folder where query result ZIP files are written.**	**C:\SCG\QrFiles**
QueryResult:MaxMailAttachmentMB	Largest email attachment size. Larger files are not attached; the user downloads them from the interface.	10
Reports:QueryTimeoutSeconds	Timeout for report queries.	500
RateLimiting:PermitPerMinute	Requests allowed per minute.	60
RateLimiting:QueueLimit	Requests held in the queue.	10
Workers:ApiKey	The key the worker uses to identify itself to the API. Must match WorkerApi:Key on the worker side.	REPLACE_IN_PROD
Workers:WorkerErrorMailTo	Address that receives notifications when the worker hits an unexpected error.	
Workers:StaleExecutionRecoveryMinutes	Crash recovery threshold in minutes. An execution running longer than this is presumed dead and the request is returned to Approved. See Section 25.7.	120

The worker service account must be able to write to QueryResult:OutputPath. Without permission the query runs but the file cannot be created, and a packaging error appears in the request detail.

2.5 Worker side appsettings.json

The worker file is short:

Key	**What it does**
WorkerApi:BaseUrl	API address. The worker reports its status here.
WorkerApi:Key	Must match Workers:ApiKey on the API side.
Serilog:MinimumLevel	Log detail level.

If the two keys do not match, the worker runs but cannot report status. It appears offline in the interface.

2.6 First administrator and first sign-in

If there is no user at installation time, a single administrator account is created:

Field	Value
Username	admin
Initial password	admin123
Role	SystemManager (35)

Change this password right after the first sign-in. This account ships alone in the production copy; no other sample user is created.

2.7 Installation verification checklist

The installation is successful if all of the following hold:

[ ] The API address opens in a browser and the sign-in screen appears.

[ ] Sign-in with admin works and the password has been changed.

[ ] Settings > License shows a valid license status.

[ ] Settings > Servers has at least one server and its connection test succeeds.

[ ] The checks on Settings > Diagnostics are green.

[ ] The worker service is running and shows as online in the interface.

[ ] A test request can be opened and approved.

[ ] The Audit Log screen shows records for these actions.

2.8 Common installation problems

Symptom	Likely cause	Fix
API does not start, connection error in the log	ConnectionStrings:Default is wrong or the database does not exist	Check the connection string and database access.
Screens open but every request is unauthorised	Jwt:SigningKey changed while a session was open	Sign out and sign in again.

Symptom	Likely cause	Fix
The web interface cannot reach the API	The interface address is missing from AllowedOrigins	Add the address and restart the API.
Write operations return 402	The license is expired or invalid	Upload a valid license. See Section 3.6.
Email is not delivered	Mail:Enabled is false or DevMode is true	Fix the settings and follow the outcome on the Notification Logs screen.
A scheduled request does not run	The worker has stopped or MainWorkEnabled is false	Start the service and set the flag to true.
The query result file is not created	QueryResult:OutputPath does not exist or is not writable	Create the folder and grant write permission to the service account.
Server password cannot be decrypted	Security:ServerPasswordKey changed	Restore the old key or re-enter the server passwords.

Section 3. Licensing

3.1 What the license file is

The license is a signed file with the .scglicense extension. It contains the customer name, validity dates, enabled modules, the allowed number of servers and the allowed database types.

The file is digitally signed. The signature is verified with the public key in the License:PublicKeyPem setting. A file whose content has been edited by hand fails signature verification and is treated as invalid.

3.2 License content

Field	What it does	If left empty
CustomerName	Customer name. Shown in the interface.	-
Edition	Edition name. Informational only.	Core
MaxInstances	Maximum number of managed servers allowed. All rows in the Servers table are counted, including inactive ones.	0 means unlimited
DatabaseTypes	Allowed target database types.	Empty list = all enabled
Modules	Enabled modules.	Empty list = all enabled
GracePeriodDays	How many days the system keeps working after expiry.	7
ValidFrom, ValidTo	Validity range.	-
IssuedAt	Issue date.	-

A rule to watch: an empty Modules list means "all enabled". To state that no optional module is enabled, the single value None is written into the list. Confusing these two would enable every module in a core-only sale.

3.3 Modules

There are six optional modules. Core functions (opening requests, approval flow, execution, audit trail, user and role management, LDAP sign-in) are always available regardless of modules.

Module	Features it enables	What happens when disabled
QueryGovernance	Opening query result requests, sensitive data masking, encrypted result delivery, result file download.	Query result requests cannot be saved or executed and the download button is hidden.
Sandbox	Trying a request on a test server.	The sandbox button is hidden and the sandbox command is rejected.

Module	Features it enables	What happens when disabled
RollbackRecovery	Automatic rollback script generation, per-server rollback requirement, Oracle automatic rollback.	The generate rollback button is hidden. In addition the "Rollback Required" rule stops being enforced: with no automatic generation available, the requirement is not applied so that execution is not locked. The flag on the server definition stays visible but no longer blocks execution.
ScheduledExecution	Scheduling a request for a later time, per-server automatic execution.	The schedule button is hidden and no automatic scheduling happens after approval.
ReleasePackages	Creating and running release packages.	Actions on the release package screen are rejected.
InsightsReporting	Management reports and the Object History screen.	Reports and object history do not open.

An important principle: historic data of a disabled module stays readable. For example if the release package module is switched off, existing packages remain visible in the list; only new actions are blocked. Data is never held hostage.

When a module is disabled the related button is hidden, not greyed out. The reason is that letting a user try an action they can never perform, only to receive an error, has no value.

3.4 Server limit

MaxInstances is compared against the number of rows in the Servers table. Inactive servers are counted too. When the limit is reached no new server can be added; existing servers keep working.

Example: with MaxInstances set to 3 and three servers defined, adding a fourth fails. Deactivating a server does not free a slot; it has to be deleted.

3.5 Database types come from the license

The database type list on the server definition screen is populated from the license. If DatabaseTypes is empty in the license, all three types can be selected. If, for example, only SqlServer is present:

- Only SQL Server appears in the list when adding a new server.

- If an Oracle server was added earlier, execution, connection testing and sandbox operations on that server are blocked. The record is not deleted, it simply cannot be used.

3.6 License states and system behaviour

State	Meaning	System behaviour
Valid	Valid.	Everything works normally.
Trial	Trial license.	Everything works, all modules are treated as enabled.
Expiring	Fewer than ExpiryWarningDays days remain.	Everything works, a warning appears in the interface.
GracePeriod	Expired but inside the grace period.	Everything works, a warning appears.
Expired	Validity and grace period are over.	Restricted mode.
Invalid	Signature could not be verified or the file is corrupt.	Restricted mode.
Missing / NoLicense	No license file.	Restricted mode.

What restricted mode means: all write operations (POST, PUT, PATCH, DELETE) are rejected with code 402 and the message "license invalid or expired". Three endpoints are excepted: sign-in operations, license operations and the API documentation. So you can sign in and upload a new license, but you cannot create or change any other record. Read operations continue to work and you can still see your data.

3.7 Uploading a license

1. Open the Settings screen.

2. Go to the License tab. Your role must hold the license management permission to see this tab.

3. The current status card shows the customer name, expiry date and remaining days.

4. Select the license file or paste its content.

5. Save. The application verifies the signature and reloads the license.

If the upload succeeds the status card updates immediately. No restart is required.

3.8 License error messages

Message	Cause	Fix
License invalid or expired	A write attempt in restricted mode	Upload a valid license.
Signature could not be verified	The file was modified or belongs to another organisation	Request the correct file from your supplier.

Message	Cause	Fix
Server limit exceeded	MaxInstances is full	Delete an unused server definition or upgrade the license.
This module is not covered by your license	An action of a disabled module was called	Have the module added to your license.

Section 4. Roles, Permissions and Hierarchy

4.1 Roles and their numeric values

There are eight roles. Role ids increase in steps of five.

Id	Role name	Hierarchy level	Short description
0	NoAccess	0	No access. The account exists but can do nothing.
5	Viewer	5	Sees only the dashboard.
10	Auditor	10	Observes, produces evidence and verifies it. Cannot enter the change flow.
15	Requester	15	Opens requests.
20	RequestManager	20	Opens and approves requests.
25	DbAdmin	25	Database administrator. Executes and manages all settings.
30	ChangeManager	30	Change manager. Holds the same permissions as DbAdmin and sits above it in the hierarchy.
35	SystemManager	35	System manager. The highest level. Its role definition cannot be edited.

Why steps of five: if a new role is ever needed *between* two existing roles, one of the free numbers in between is used and no user record has to be shifted. With consecutive numbering (between 4 and 5) that would be impossible.

4.2 Id and hierarchy are separate concepts

This is the item technical teams should read most carefully.

Every role carries two numbers:

- Id: the number stored on user records.

- Level: the number that places it in the hierarchy.

Today these hold the same values, but they are not the same concept. A role can be appended at the end of the id list yet placed in the middle of the hierarchy. The Auditor role is managed separately for exactly this reason: its observation rights are wider than Viewer, yet it has no authority at all in the change flow.

Rule: role comparison is never made by looking directly at the enum number. Comparison always goes through the level. Breaking this rule silently grants authority to a newly added role.

If an unknown role value arrives, the level is treated as 0. So an undefined role gains nothing.

4.3 The special position of the Auditor role

The auditor cannot be selected as an approver in the approval flow. If the person who audits also approves, segregation of duties collapses, because they end up auditing work they approved themselves.

What the auditor can do: view reports, view the dashboard, view audit records, view all requests, view notification logs, export evidence dossiers.

What the auditor cannot do: open requests, approve, execute, run sandbox, change any setting, manage release packages.

4.4 Role permission matrix

The table below shows the default role definitions. They can be edited on the Roles screen; the SystemManager role cannot be edited.

Permission	NoAccess	Viewer	Auditor	Requester	RequestManager	DbAdmin	ChangeManager	SystemManager
Add request	-	-	-	Yes	Yes	Yes	Yes	Yes
Execute	-	-	-	-	-	Yes	Yes	Yes
Approve	-	-	-	-	Yes	Yes	Yes	Yes
Sandbox	-	-	-	-	-	Yes	Yes	Yes
View reports	-	-	Yes	-	Yes	Yes	Yes	Yes
View dashboard	-	Yes	Yes	Yes	Yes	Yes	Yes	Yes
View audit log	-	-	Yes	-	-	Yes	Yes	Yes
Manage release packages	-	-	-	-	-	Yes	Yes	Yes
View all requests	-	-	Yes	Yes	Yes	Yes	Yes	Yes

Permission	NoAccess	Viewer	Auditor	Requester	RequestManager	DbAdmin	ChangeManager	SystemManager
Manage servers	-	-	-	-	-	Yes	Yes	Yes
Manage users	-	-	-	-	-	Yes	Yes	Yes
Manage roles	-	-	-	-	-	Yes	Yes	Yes
Manage parameters	-	-	-	-	-	Yes	Yes	Yes
Manage approval policies	-	-	-	-	-	Yes	Yes	Yes
Manage SQL standards	-	-	-	-	-	Yes	Yes	Yes
Manage critical objects	-	-	-	-	-	Yes	Yes	Yes
Manage ITSM tickets	-	-	-	-	-	Yes	Yes	Yes
Manage sensitive columns	-	-	-	-	-	Yes	Yes	Yes
Manage sensitive patterns	-	-	-	-	-	Yes	Yes	Yes
Manage email sources	-	-	-	-	-	Yes	Yes	Yes
Manage maintenance	-	-	-	-	-	Yes	Yes	Yes
View notification logs	-	-	Yes	-	-	Yes	Yes	Yes
Manage license	-	-	-	-	-	Yes	Yes	Yes
Use diagnostics	-	-	-	-	-	Yes	Yes	Yes

Permission	NoAccess	Viewer	Auditor	Requester	RequestManager	DbAdmin	ChangeManager	SystemManager
Export evidence dossier	-	-	Yes	-	-	Yes	Yes	Yes

Roles that can add requests can also open query result requests.

4.5 Permission keys

Every permission is represented by a key which is written into the sign-in token. Screen and button visibility reads these keys.

Key	Action it enables
cr.add	Opening change and query result requests
qr.add	Opening query result requests
cr.execute	Executing a request
cr.sandbox	Running sandbox
cr.approve	Approving, rejecting, scheduling, marking manual
cr.seeall	Seeing other people's requests
report.view	Reports screen
dashboard.view	Dashboard
audit.view	Audit log screen
audit.evidenceexport	Exporting an evidence dossier
rp.manage	Managing release packages
settings.servers and the other settings.* keys	The related settings tab

4.6 Hiding in the interface is not security

Hiding a button does not prevent a user from performing the action. The web interface runs in the browser and a user can send a request straight to the API.

For this reason every action is checked twice:

1. The interface decides whether to show the button.

2. The API applies the same rule independently when it receives the request.

The second check is the real security. The first exists only for user experience.

4.7 User lifecycle

Action	Behaviour
Adding a user	Username and email must be unique.
Deleting a user	The record is not physically deleted, it is marked as deleted (soft delete). The name is preserved on historic requests.
Adding a user with the name of a deleted one	No new record is created; the old record is revived. The same person returns to the same history.
Deleting the last system manager	Blocked. At least one system manager must remain.
Deleting yourself	Blocked.
Four eyes deadlock gate	While settings approval is on, deleting the second-to-last holder of a screen permission is blocked. Otherwise changes on that screen could never be approved.
Account lock	Consecutive failed sign-ins lock the account temporarily. An administrator can also unlock it.

When a permission change takes effect

When a user's role changes, their account is closed, or the role permission matrix is updated, the change takes effect without delay. Two mechanisms deliver this together.

1. High impact operations read the permission from the database. In the operations below the permission is verified directly against the database, not against the session key the user holds:

- Executing a request

- Scheduling an execution

- Approving a request

- Approving and rejecting a settings change

A user whose permission has been withdrawn, or whose account has been closed, cannot perform these operations even while their session still appears open.

2. Session renewal is closed. When a role changes, an account is closed, or a role permission matrix is updated, the session renewal records of the affected users are revoked. Once the session key they hold expires, renewal fails and the user has to sign in again.

The revocation is written to the audit trail as well: User.SessionsRevoked for a per user revocation and Role.SessionsRevoked for a per role one. The entry carries the reason and how many sessions were closed.

A newly granted permission becomes visible after the user signs in again. This is deliberate: withdrawing a permission takes effect immediately, granting one takes effect when the session is renewed.

4.8 LDAP users versus local users

Topic	Local user	LDAP user
Where the password lives	In the SQL Change Guard database, as an encrypted digest.	In the corporate directory. The application never stores it.
Changing the password	Inside the application.	In the corporate directory.
Role	Assigned inside the application.	Assigned inside the application. Directory group membership does not decide the role.
Account lock	Managed by the application.	The directory may also lock it; the application lock is separate.

4.9 Password policy and BCrypt

What a hash is

Turning a text irreversibly into a fixed-length value is called hashing. Passwords are not stored as plain text; their digests are. When a user signs in, the digest of the entered password is computed and compared with the stored digest.

Someone who steals the database sees digests, not passwords.

What a salt is

If two users with the same password produced the same digest, cracking one would crack the other. To prevent this a random extra value (a salt) is mixed into each password, so the same password produces a different digest for different users.

Why BCrypt was chosen

SQL Change Guard uses BCrypt for user passwords.

Property	Plain SHA-256	BCrypt
Speed	Very fast. Billions of attempts per second are possible.	Deliberately slow.
Salt	Must be added separately.	Generated and stored internally.

Property	Plain SHA-256	BCrypt
Adjustable cost	None.	Yes (work factor).
Suitable for storing passwords	No.	Yes.

What the work factor is: the number that decides how many rounds BCrypt performs. Increasing it by one doubles the time. As hardware gets faster this number is raised to keep the same protection level.

Being fast is normally a good property for SHA-256, but for password storage it is a weakness: the attacker also gets that speed. BCrypt is deliberately slow; a delay a user never notices at sign-in becomes an unaffordable cost for someone trying billions of guesses.

SHA-256 is used elsewhere in the product for a different purpose: the audit chain. There the goal is not to store a password but to prove that content has not changed, and speed is not a weakness. See Section 5.4.

4.10 The Users screen

Screen: Settings > Users

Permission: settings.users

Field	What it does	Required	Notes
Domain Username	The sign-in name of the user. This name appears on requests, approvals and audit records.	Yes	It is unique. Entering the name of a deleted user revives the old record.
Email	The address notifications are sent to.	Yes	It is unique. Also used in query result email approval.
Role	The role of the user.	Yes	Section 4.1.
Manager	The direct manager of the user.	No	If they meet the role of an approval step, the step is assigned to them and the notification goes to them first.
Authentication Type	Local account or directory account.	Yes	Section 4.8.
Password / New Password	The password for local accounts.	Yes for local accounts	Hidden for directory accounts. The password policy comes from the settings in Section 2.4.3.

Field	What it does	Required	Notes
Locked	Whether the account is locked.	-	Set automatically after failed sign-in attempts.

Actions on the screen:

Action	What it does
Unlock	Opens a locked account before the lock period ends.
Reset password	Assigns a new password to a local account.
Import CSV	Adds users in bulk. Every row in the file is validated individually and faulty rows are reported in the result summary.
Delete	Performs a soft delete. The protection rules in Section 4.7 apply.

While the four eyes rule is on, changes on this screen also go for approval. See Section 8.

4.11 The Roles screen

Screen: Settings > Roles

Permission: settings.roles

Each permission listed in Section 4.4 can be switched on or off per role. Three rules apply:

1. The SystemManager role cannot be edited. Otherwise someone could restrict the top role and leave the system unmanageable.

2. The auditor role cannot be granted approve or execute permissions. This follows from segregation of duties.

3. The hierarchy level of a role is not editable from the screen. The level is part of the product definition.

Section 5. Security and Encryption Model

5.1 Three different needs, three different methods

Need	Method used	Why
Storing user passwords	BCrypt	Must not be reversible, must be expensive to crack.
Storing server passwords and result package passwords	AES-256-GCM	The application must be able to read these values back.
Proving the audit trail has not changed	SHA-256 and HMAC-SHA256	Reading back is not needed, only verification.

The basic distinction: if you need to read a value back, use encryption; if you only need to verify it, use a digest.

5.2 Server passwords: AES-256-GCM

What symmetric encryption is

A method where the same key both encrypts and decrypts is called symmetric encryption. AES is the most common example. The 256 is the key length in bits.

Why GCM was chosen

AES has several modes of operation. GCM mode produces an authentication tag in addition to the ciphertext. If the ciphertext is altered later, decryption fails instead of silently returning corrupted data.

In classic modes (CBC for example) an attacker can tamper with the ciphertext and the application may produce a meaningless result without noticing. GCM removes this risk.

Where it is used

Data	Where it is stored	Key
Database password in a server definition	Servers.SqlPassword column	Security:ServerPasswordKey
Query result ZIP password	ChangeRequests.QrZipPassword column	Security:QrZipPasswordKey

The keys live in configuration, not in the database, so someone with database access alone cannot decrypt these values.

If the key changes: old values cannot be decrypted. Server passwords must be re-entered. Key rotation must therefore be planned.

5.3 What is protected where

Data	State
User password	Stored as a BCrypt digest. Not readable.
Server database password	Stored encrypted with AES-256-GCM. Never shown in any screen.
Query result ZIP password	Stored encrypted with AES-256-GCM. Shown only to authorised users.
Query result data	Held on disk inside an encrypted ZIP, deleted when the retention period ends.
Audit trail	Stored as plain text but signed with a chain.
Log files	Password and secret fields are masked.

5.4 Audit trail: SHA-256 and HMAC

What SHA-256 is

An algorithm that produces a fixed 64-character digest from a text. The same text always gives the same digest; changing a single character changes the digest entirely. Recovering the text from the digest is practically impossible.

Why a hash chain provides immutability

When the digest of an audit record is computed, the digest of the previous record is included. Records are therefore linked:

```
Record 1: content1 + "GENESIS"  --> digest1
Record 2: content2 + digest1    --> digest2
Record 3: content3 + digest2    --> digest3
Record 4: content4 + digest3    --> digest4
```

If someone edits record 2, digest2 no longer matches. Once digest2 changes, digest3 no longer matches, and once digest3 changes, digest4 no longer matches. Quietly fixing a single record therefore requires recomputing every record after it.

The difference between HMAC and a plain digest

With plain SHA-256 the formula is public. Someone with write access to the database can change records and recompute the whole chain, because no secret is needed.

With HMAC-SHA256 a secret key enters the computation. The key is not in the database, it comes from configuration. Someone who does not know the key cannot produce a valid digest and cannot recompute the chain.

Versions

Version	Algorithm	Scope
1	SHA-256	Old value and IP address excluded. Seen only in legacy records.
2	SHA-256	Old value and IP address included. Written when no HMAC key is configured.
3	HMAC-SHA256	Same scope as version 2, plus a secret key.

When a key is configured, new records are written as version 3. Each record also stores which key signed it (KeyId), so old records remain verifiable after a key change.

A version 3 record whose key is missing from the ring counts as "not verified". That is always the safe side.

5.5 Database level write log

Alongside the audit trail a second record is kept: TableWriteLogs. Whenever a row is inserted, updated or deleted in a watched table, its old and new state is written here. Database triggers write it.

The two serve different purposes:

	Audit trail (AuditLogs)	Write log (TableWriteLogs)
What it holds	Governance events that went through the application	Raw write activity in the database
Who writes it	The application	A database trigger
Sees changes that bypass the application	No	Yes
Seal	HMAC, key kept outside the database	None, see below

The write log also stores the database login that made the change. If someone alters a record directly with SQL, the login responsible is visible here.

Why the write log has no seal

The audit trail is sealed with HMAC and the key is kept outside the database, so someone with database access alone cannot recompute it. That is where the audit evidence lives.

The same method is not workable for the write log: the seal would have to be computed inside the database and the key would have to live there too. Someone with write access to the database, that is the very person the seal is meant to catch, could then delete a record and recompute the seal. Instead of a seal that provides no protection, two real controls are used.

What to do at installation

1. Grant insert only on the write log. The application account never needs to modify or delete this table:

```
GRANT INSERT ON dbo.TableWriteLogs TO [application_account];

DENY UPDATE, DELETE ON dbo.TableWriteLogs TO [application_account];
```

This step is left to installation because the account name differs per organisation. Once applied, corrupting the write log through the application becomes impossible.

2. Keep the daily anchor running. Every day the worker seals a scope summary of the write log (row count and id range) into a separate database. Bulk deletion is therefore detectable from outside the main database. This happens on its own as long as the worker runs.

The one case left is someone with update rights editing a single row in place. No mechanism inside the database can prevent that; the answer is the permission step above.

5.6 Secret masking

Fields such as passwords, secrets and connection strings are masked in what is written to the audit trail and to logs. The purpose is that someone allowed to read the audit trail cannot harvest secrets from it.

Section 6. Server Definitions

Screen: Settings > Servers

Permission: settings.servers

6.1 What this screen is for

Target database servers on which requests and queries will run are defined here. Without a server definition no request can be opened, because every request must select at least one target server.

6.2 Field by field

Field	What it does	Required	Valid values	Default	If left empty	Effect of changing it
Server Name	The name identifying this server in the system. Requests and reports show this name.	Yes	Free text	-	Not saved, error	Historic requests store the server name as text, so past records keep the old name.
Host	Network address or name of the server.	Yes	Name or IP	-	Not saved	New connections go to the new address.
Port	Connection port. Updated automatically when you change the database type.	No	Number	SQL Server 1433, Oracle 1521, PostgreSQL 5432	Default is used	
Database Type	Type of the target. The list narrows according to the license.	Yes	SQL Server, Oracle, PostgreSQL	SQL Server	-	Script parsing and execution behaviour change completely.
Database Name	The database to connect to. Required for PostgreSQL. For Oracle it is the service name, usually supplied through the host.	Depends on type	Free text	Empty	The default database is used on SQL Server	

Field	What it does	Required	Valid values	Default	If left empty	Effect of changing it
Environment	Which environment the server belongs to. Directly affects approval policy selection.	Yes	Production, Staging, Development, Test	Production	-	Policy matching changes and new requests may get different approval steps.
SQL Username	The account used to connect to the target.	Yes	Free text	-	Connection fails	
SQL Password	The account password. Stored encrypted and never displayed again.	Yes on new records	Free text	-	Leaving it blank while editing keeps the existing password	
Active	Whether the server is in use.	-	On / Off	On	-	If off, it cannot be selected on new requests.
CR Auto Execution	Should approved change requests run automatically on this server. Requires the scheduling module.	-	On / Off	Off	-	See 6.4
QR Auto Execution	Should approved query result requests run automatically on this server.	-	On / Off	Off	-	See 6.4
Rollback Required	Execution on this server requires an existing rollback script.	-	On / Off	Off	-	When on, requests without a rollback cannot be executed.

Field	What it does	Required	Valid values	Default	If left empty	Effect of changing it
Auto Rollback	Should a rollback script be generated automatically when the request is approved.	-	On / Off	Off	-	
SSL Required	Visible for PostgreSQL only. Required by cloud PostgreSQL services.	-	On / Off	Off	-	
Masking	How sensitive data detection works for query results from this server.	Yes	Full (catalog + pattern), Content only	Full	-	See 6.5
Exemption reason	A written reason when masking is not set to Full.	Yes unless Full	Free text	Empty	Not saved, error	Written to the audit trail.

6.3 Connection test

The server edit screen has a connection test button. It tries to connect with the entered details and shows the result immediately.

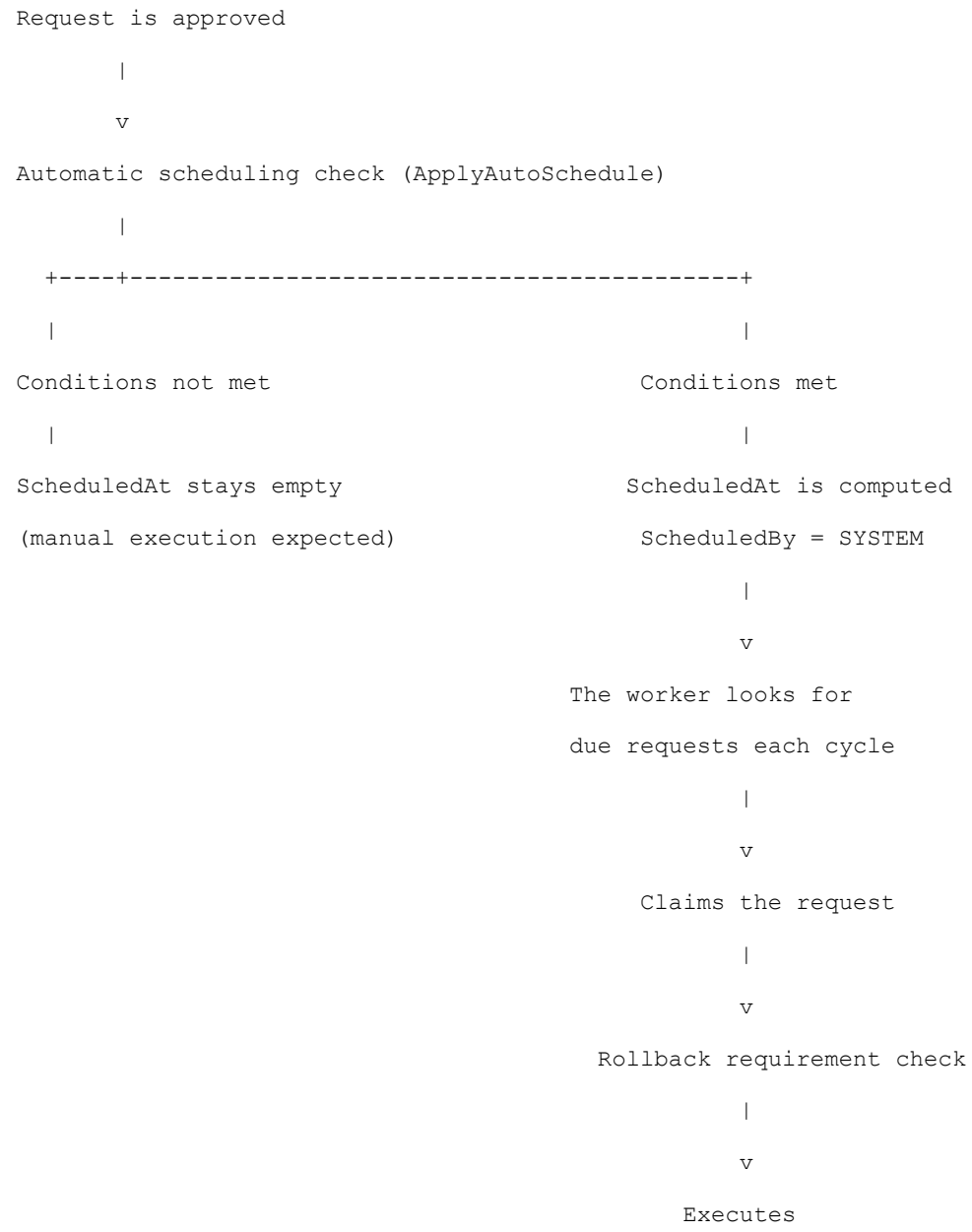
Result	Meaning	What to do
Success	The connection was established.	-
Authentication error	Username or password is wrong.	Check the credentials.
Server unreachable	Network, host name or port is wrong, or a firewall is blocking.	Check network access.
Database not found	The database name is wrong.	Check the name.
This database type is not covered by your license	The license does not allow this type.	See Section 3.5.

The worker also runs ConnectionCheckJob at regular intervals, trying to connect to every active server and updating the status. This makes an unreachable server visible before a request runs.

6.4 Automatic execution: what happens behind the scenes

This is one of the most frequently asked topics in this guide. Step by step:

The chain



The conditions

Automatic scheduling happens only if all of the following are true:

1. The scheduling module is licensed.

2. None of the risk findings on the request is flagged never-skip.

3. The request is not a rollback request.

4. The request is not inside an active release package.

5. The relevant auto execution flag is on for every target server. Change requests look at the CR flag, query result requests at the QR flag.

6. If it is a change request and any server has "Rollback Required" on, a rollback script must already exist.

If any condition fails, the request stays approved and waits for someone to execute it manually. Any schedule previously set by the system is cleared.

How the time is computed

The order is:

1. If the user supplied a specific execution date on the request (RequestedExecutionDate), that is used.

2. If not, and a delay in minutes was supplied (ScheduleDelayInMinutes), the approval time plus that delay is used.

3. If neither, the approval moment itself is used.

Then the ExecutionDelaySeconds parameter applies: if the computed time is earlier than "now plus ExecutionDelaySeconds", it is pushed to that value.

Concrete example: ExecutionDelaySeconds is 300 (5 minutes). You approve a request at 14:00 and the request carries no specific date.

- Computed time: 14:00

- Earliest allowed time: 14:05

- Result: the execution time becomes 14:05.

The worker does not pick the request up before 14:05. This waiting period is deliberate: it leaves human time to cancel a request after an approval given by mistake.

The same parameter also acts as a lower bound for manual scheduling, but there a minimum of 30 seconds applies: even with ExecutionDelaySeconds set to 10, manual scheduling cannot pick a time earlier than 30 seconds from now.

Who the record names

In automatic scheduling nobody pressed a button. Therefore:

- ScheduledBy = SYSTEM

- ExecutionRequestedBy = SYSTEM

If the field were left blank, whoever reads the report would wonder whether it was forgotten. The owner of the decision is written explicitly as SYSTEM.

6.5 Masking regime

Regime	What it does	When to use it
Full (Strict)	Looks at both the classification catalog (column name) and the data patterns.	Any environment holding real data. This is the default.
Content only (ContentOnly)	Ignores column names and inspects values only.	Test environments holding synthetic (generated) data.

There is deliberately no "off" option at server level. Switching masking off entirely is possible only through the global QrMaskingEnabled parameter, and that counts as a control change.

The content only regime verifies itself: in this regime the column name is ignored, but value-based checks keep running. If real data is loaded onto the server, fields such as national identity numbers, card numbers and IBANs are still caught and masked. In other words the exemption opened "because it is a test environment" narrows itself back into protection the moment real data appears.

What stays open in this regime: names, addresses, diagnoses and similar information that has no verifiable pattern. The interface shows this warning explicitly.

When a regime other than Full is chosen a written reason is mandatory, and it is frozen onto the delivery record. Even if the server is later switched back to Full, the regime used for that delivery remains in the evidence.

6.6 The multi-server rule

A request can target several servers. Server names are stored in a single field, separated by semicolons:

```
PROD-SQL01;PROD-SQL02;PROD-SQL03
```

The consequences:

- The server column in lists and reports may contain several names.

- Approval policy resolution runs separately for each server and the strictest outcome applies.

- For automatic execution the flag must be on for all servers.

- In query result requests each server is evaluated with its own masking regime; a production server can be masked while a test server stays open.

- Reports filtering by server name search inside the list rather than matching exactly.

6.7 Deleting versus deactivating a server

Action	Effect
Deactivating	Cannot be selected on new requests. Existing requests are unaffected. It still occupies a slot in the license count.
Deleting	The record leaves the table and a license slot is freed. Historic requests keep the server name as text, so records are not broken.

Section 7. Parameters

Screen: Settings > Parameters

Permission: settings.params

7.1 What this screen is for

Single settings that shape system behaviour are held here. Each parameter has a key, a value and a group. The screen lists parameters by group.

7.2 Fields on the screen

Field	Meaning
Key	The system name of the parameter. Not editable.
Value	The effective value.
Description	What the parameter does.
Control type	How the value is entered: Numeric, Boolean, Text, Combo.
Options	Valid values for a Combo parameter.
Group	The heading on the screen.
Control setting badge	Shown when the parameter directly determines the strength of a control. See Section 8.

7.3 General group

DefaultQueryTimeout

- What it does: the starting value and the lower bound of the query timeout field on new requests.

- Value: number of seconds.

- Default: 30

- Where it is read: the request editor screen and the save request command.

- Example: with 30, a new request opens with a timeout of 30. If you type 10, it is raised back to 30 when saving.

- If empty or 0: 30 is assumed.

- Does it affect history: no, it applies to new requests only.

MaxQueryTimeout

- What it does: the highest timeout that can be given to a request.

- Value: number of seconds.

- Default: 3600 (1 hour)

- Where it is read: the save request and save release package commands.

- Example: with 3600, a user entering 7200 is lowered to 3600.

- Why it exists: a very long-running script holds locks on the target server and blocks other work. An upper bound limits that risk.

ExecutionDelaySeconds

- What it does: the minimum time between approval and execution.

- Value: number of seconds.

- Default: 300 (5 minutes)

- Where it is read: the approve command, the approval workflow service and the manual schedule command.

- Example: described in detail in Section 6.4. With 300, a request approved at 14:00 runs at 14:05 at the earliest.

- If 0: no waiting is applied and automatic execution starts at the moment of approval. A 30-second lower bound still applies to manual scheduling.

- Why it exists: it leaves time to reverse an approval given by mistake.

RejectOnError

- What it does: should a request be rejected automatically when execution fails.

- Value: true / false

- Default: true

- Where it is read: the scheduled execution service.

- Example: with true, a request that fails overnight becomes Rejected and is not retried. With false it returns to Approved and can be retried.

- Control setting: yes.

- Which to choose: organisations with strict controls prefer true; a failed job should not be retried silently.

7.4 Segregation of Duties group

Parameters in this group are organisation-wide defaults. An approval policy can override them per server or environment. The effective values are frozen onto the request when the approval flow is initialised; later changes never rewrite an already decided request.

RequesterCanSelfApproveExecute

- What it does: may the person who opened a request approve and execute it themselves.

- Value: true / false

- Default: true

- Where it is read: approve, execute, schedule and mark-manual checks.

- Example: with false, a developer trying to approve their own request gets "the requester cannot approve their own request" and the button appears disabled.

- Control setting: yes.

ApproverCanExecute

- What it does: may the person who approved a request also execute it.

- Value: true / false

- Default: true

- Where it is read: execute and schedule checks.

- Example: with false, the manager who approved sees the execute button disabled; another authorised user must run it.

- Why it exists: separating the approver from the implementer is a common audit expectation.

- Control setting: yes.

RequireDistinctApproverPerStep

- What it does: may the same person close more than one approval step on a request.

- Value: true / false

- Default: false

- Example: in a two-step policy, with false a single person holding the right role can close both steps. With true, a different person is needed for the second step. That is the genuine four eyes rule.

- Control setting: yes.

EmergencyMinimumRole

- What it does: the lowest role allowed to raise an emergency request.

- Value: role name.

- Default: Requester

- Example: setting it to ChangeManager stops developers from raising emergency requests; only change managers and above can.

- Why it exists: declaring an emergency is an authority. If everyone holds it, the normal approval path is bypassed easily.

- Control setting: yes.

7.5 Settings Approval group

SettingsApprovalPolicy

- What it does: the scope of the four eyes rule on settings screens.

- Values:

  - Disabled: changes are applied immediately.

  - ControlSettingsOnly: only changes that weaken or strengthen a control wait for approval.

  - All: every change on those screens waits for approval.

- Default: Disabled

- Control setting: yes. Switching it off also requires approval; switching it on applies immediately, because while the policy is off the interception does not run anyway.

- Details: Section 8.

7.6 ITSM Ticket group

ChangeTicketRequired

- What it does: is a ticket number mandatory when opening a request.

- Value: true / false

- Default: true

- Example: with true, a request cannot be saved without a ticket number.

- Control setting: yes.

ChangeTicketRegex

- What it does: format validation of the ticket number.

- Value: a regular expression. Can be left empty.

- Default: ^CR-\d{5}$

- Example: with the default, CR-12345 is accepted while 12345 and CR-123 are rejected.

- If empty: no format check is applied and any text is accepted.

- For your own organisation: if your ticket numbers look like INC0001234, use ^INC\d{7}$.

7.7 Query Result group

QrFileRetentionDays

- What it does: how many days result ZIP files are kept on disk.

- Value: number of days.

- Default: 1

- Where it is applied: the maintenance task (QR File Cleanup).

- Example: with 1, a file produced today is deleted tomorrow. The deletion is written to the audit trail and appears in the Data Access report.

- Control setting: yes.

- Why it is kept short: the shorter a file containing live data stays on disk, the smaller the risk.

QrRequesterCanDownload

- What it does: may the person who opened the request download their own result file.

- Value: true / false

- Default: true

- Example: with false, the person who asked for the data cannot download the file; only those above the role threshold below can. This is for organisations that want to separate the person requesting data from the person accessing it.

- Control setting: yes.

QrDownloadMinimumRole

- What it does: the download threshold for people other than the requester.

- Value: role name.

- Default: ChangeManager

- Example: with ChangeManager, a DbAdmin cannot download someone else's result file because they sit lower in the hierarchy.

- Control setting: yes.

7.8 Sensitive Data group

QrMaskingEnabled

- What it does: the master switch for masking.

- Value: true / false

- Default: true

- If switched off: no detection and no masking happens. Every delivery is still recorded with the applied policy shown as MaskingOff. This is the only place where an auditor can see a result sent while masking was off.

- Control setting: yes.

QrSensitiveDataPolicy

- What it does: the behaviour while masking is on.

- Values:

  - AutoMask: detected sensitive columns are masked. Approved exceptions stay open.

  - WarnOnly: nothing is masked, the detection result is only recorded.

- Default: AutoMask

- Example: WarnOnly is used briefly on a new installation to see whether the patterns behave correctly. Using it permanently removes the protection.

- If an unknown value is written: the safe side wins and masking is applied.

- Control setting: yes.

QrSensitiveDetectionSampleSize

- What it does: how many sample rows are taken per column for pattern scanning.

- Default: 1000

- Example: with 1000, in a 500,000-row result 1000 cells per column are inspected. Raising the number improves accuracy and lengthens the run.

- Control setting: yes.

QrSensitiveDetectionThresholdPct

- What it does: what percentage of sampled cells must match for a column to count as sensitive.

- Value: 0-100.

- Default: 20

- Example: with 1000 rows sampled and a threshold of 20, finding the identity number pattern in 200 cells marks the column sensitive and it is masked. Finding it in 150 cells does not.

- If lowered: more columns are masked, false positives increase.

- If raised: fewer columns are masked, the risk of missing one increases.

- Control setting: yes.

QrSensitiveDetectionSampleMode

- What it does: how the sample rows are chosen.

- Values: Random, Sequential

- Default: Random

- Example: in sorted data the first 1000 rows may all come from the same region and not be representative. Random spreads the selection across the whole result.

- Control setting: yes.

QrSensitiveDetectionTimeBudgetSeconds

- What it does: the total time budget for the whole detection pass.

- Default: 10

- If the budget is exceeded: the remaining columns are marked "not scanned" and are masked. The safe side always wins.

- If 0: there is no limit.

- Control setting: yes.

QrColumnPreviewEnabled

- What it does: should the column list be fetched from the target server while a query result request is saved.

- Default: true

- What it does technically: it does not run the query. It only asks which columns the query returns and reads no rows.

- Why it matters: the preview learns the real source of aliased columns. Writing SELECT TCKN AS Code shows a column named Code while the source column is TCKN. Without this information masking could be bypassed.

- If the server cannot answer: the request is still saved.

UnmaskedColumnRequestPolicy

- What it does: may a user ask for specific columns to be delivered unmasked.

- Values:

  - RequireApproval: the requested columns are listed and an extra approval step is added. Approved columns are not masked.

- AllowWithReason: no extra step; a reason is written and it counts as auto-approved.

- Disabled: the feature is off, the related fields are hidden in the editor and any stored list is cleared.

- Default: RequireApproval

- Control setting: yes.

- Dependency: if QrMaskingEnabled is off, this setting behaves as Disabled. Asking for approval to unmask when nothing would be masked would turn the approval mechanism into a rubber stamp.

UnmaskedColumnApproverRole

- What it does: who approves an unmasked column request.

- Default: SystemManager

- Control setting: yes.

7.9 Email Approval group

QrEmailApprovalPolicy

- What it does: should the addresses receiving a query result go through approval.

- Values: RequireApproval, Disabled

- Default: RequireApproval

- Example: with RequireApproval, sending a result to a previously unapproved address adds an extra approval step to the request. The aim is to stop live data from quietly leaving for an external address.

- Control setting: yes.

QrEmailApproverRole

- What it does: who approves an email address.

- Default: ChangeManager

- Control setting: yes.

QrEmailAutoApproveUsers

- What it does: should addresses belonging to registered users count as approved.

- Values: True / False

- Default: True

- Example: with True, a user sending a result to their own corporate address needs no extra approval; adding an external address triggers the approval step.

- Control setting: yes.

7.10 When a parameter change takes effect

Situation	Behaviour
Newly opened requests	The new value applies immediately.
Requests whose approval flow has not been initialised	The new value applies.
Requests whose approval flow is initialised	Segregation of duties settings are frozen onto the request; the old values continue.
Completed requests	Unaffected.
Masking detection settings	The values at the time of each delivery are used and frozen onto the record.

Section 8. Settings Change Approval (Four Eyes)

8.1 What it is for

It ensures that changes on settings screens do not come from a single pair of hands. When a user changes a setting, the change is not applied immediately; it is stored as a pending record and waits for another user's approval.

Why this is needed: if a single person can switch off approval policies, the critical object list or the masking setting, the entire control system depends on that person's goodwill.

8.2 Scope

The SettingsApprovalPolicy parameter defines the scope:

Value	Scope
Disabled	No interception; every change is applied immediately.
ControlSettingsOnly	Only changes that weaken or strengthen a control become pending.
All	Every change on the covered screens becomes pending.

Covered screens: servers, users, roles, parameters, approval policies, SQL standards, critical objects, sensitive columns, sensitive data patterns, candidate columns, email sources, maintenance tasks, ITSM tickets.

What a control setting means

The parameters that become pending under ControlSettingsOnly are defined by a fixed list in code. This list is not held in the database. The reason: if the classification itself were editable, someone could first mark a parameter as "not a control" and then switch the protection off without approval.

Parameters counted as control settings:

SettingsApprovalPolicy, QrMaskingEnabled, QrSensitiveDataPolicy, QrSensitiveDetectionThresholdPct, QrSensitiveDetectionSampleSize, QrSensitiveDetectionSampleMode, QrSensitiveDetectionTimeBudgetSeconds, UnmaskedColumnRequestPolicy, UnmaskedColumnApproverRole, QrRequesterCanDownload, QrDownloadMinimumRole, QrFileRetentionDays, QrEmailApprovalPolicy, QrEmailApproverRole, QrEmailAutoApproveUsers, RequesterCanSelfApproveExecute, ApproverCanExecute, RequireDistinctApproverPerStep, EmergencyMinimumRole, RejectOnError, ChangeTicketRequired.

Approval policies, SQL standards, critical objects, sensitive columns and patterns, and the server control flags always count as control changes.

8.3 The flow

```
 User changes a setting and saves
        |
        v
 Is the policy Disabled?  ---yes---> The change is applied IMMEDIATELY
        | no
        v
 ControlSettingsOnly and the change is not a control? ---yes---> IMMEDIATELY
        | no
        v
 Does the user hold that screen permission? ---no---> The handler raises its own error
        | yes
        v
 Is there anyone else who could approve? ---no---> IMMEDIATELY (a warning is logged)
        | yes
        v
 Is there already a pending change for this record? ---yes---> ERROR: one is already
pending
        | no
        v
 A pending record is created
 The live record is NOT touched, the old value stays in effect
 "SettingChange.Requested" is written to the audit trail
        |
        v
 Another user approves or rejects
        |
   +----+------------------------------+
   |                                   |
 APPROVE                           REJECT
   |                                   |
 The command is replayed exactly   Nothing is applied
 (validation, license limits,      The requested values remain
```

```
  concurrency checks included)      in the audit trail
    |                                 |
 Success -> Approved               Rejected
 Failure -> Failed
```

8.4 What replaying the command means

When approval is given, the system retrieves the command the user originally sent and runs it again unchanged.

The benefit: validation, license limits, concurrency checks and audit writing behave at approval time exactly as they did on the first submission. This logic is not written twice, so the two paths cannot diverge.

One consequence: if conditions changed in the meantime, the command may fail. For example, if the server limit filled up meanwhile, the approval closes as "Failed" and the reason is recorded.

The created-by and modified-by fields on the record name the requester, because the change is their work. The approver's trace is kept in a separate field and in the audit trail.

8.5 Who approves

There is no separate "settings approver" role. The approver must hold the permission of that screen. A server change is approved by someone with the server permission.

An unbreakable rule: nobody can approve or reject their own request. This applies regardless of policy.

8.6 The deadlock gate

If only one person holds a screen permission, there is nobody to approve their change and it would wait forever. In this case the system applies the change directly and writes a warning to the log.

This is a deliberate choice: if the rule cannot be applied anyway, making the situation visible is better than locking the system. The lasting fix is to assign at least two users to every screen permission.

For the same reason, deleting a user also checks this condition and blocks removing the second-to-last holder.

8.7 The Pending Changes screen

Screen: Settings > Pending Changes

Column	**What it shows**
Record type	Which screen the change belongs to (server, parameter, policy and so on).
Record key	Which record will change.
Operation	Insert, update or delete.

Column	What it shows
Summary	A human-readable summary of the change.
Requested by	The user asking for the change.
Requested at	Time of the request.
Status	Pending, Approved, Rejected, Failed.
Decided by	The user who approved or rejected.
Reason	Rejection reason or failure cause.

Opening a record shows the current value next to the requested value. Fields carrying secrets such as passwords appear masked; the approval screen never opens the encrypted command payload.

A reason is mandatory when rejecting.

8.8 Messages on this screen

Message	Cause	What to do
The change was sent for approval	The record is now pending.	Ask an authorised colleague to approve.
There is already a pending change for this record	A second request was sent for the same record.	Resolve the existing request first.
A decision has already been made for this change	Someone decided before you.	Refresh the list.
You cannot approve a settings change you requested yourself	The four eyes rule.	Have another authorised user approve.
You don't have permission	You do not hold that screen permission.	Request the permission.
The change could not be applied	The command failed at approval time.	Read the reason, fix the issue and request again.

Section 9. Critical Objects, SQL Standards and Sensitive Data

9.1 Critical Objects

Screen: Settings > Critical Objects

Permission: settings.critical

What it is for

The list of database objects the organisation has declared as "touching this requires special attention". Examples: the customer table, the balance table, the authorisation procedure.

Fields

Field	What it does	Required	Example
Database Name	The database holding the object.	Yes	CustomerDB
Object Type	Table, procedure, view, function, trigger.	Yes	Table
Schema Name	The schema of the object.	Yes	dbo
Object Name	The name of the object.	Yes	Customer
Access Control	Should permission changes on this object be tracked.	-	On
Schema Change	Should structural changes to this object be tracked.	-	On

Matching logic

Matching compares four fields together: database, object type, schema, object name. The same quadruple cannot be entered twice.

Comparison is case insensitive. dbo.Customer and DBO.CUSTOMER are the same object.

What changes when a critical object is detected

1. When the request is parsed the statement is flagged as touching a critical object. The critical column in the object table of the request detail is ticked.

2. The CriticalObjectDrop standard may fire. Its severity is Critical and it is flagged never-skip.

3. The risk band rises to Critical.

4. If the approval policy has a step bound to that standard, the step is locked and cannot be skipped by any rule.

5. Automatic execution is disabled; even after approval a human must press execute.

An important rule: the criticality decision is recorded at the moment the request is parsed. Editing the catalog later does not rewrite historic requests. This is how the audit question "what did we know that day" stays answerable.

9.2 SQL Standards

Screen: Settings > SQL Standards

Permission: settings.standards

What it is for

The list of rules searched for in scripts. The installation ships with 87 standards. Each can be enabled or disabled and its severity changed.

Fields

Field	What it does	Values
Active	Should the rule run.	On / Off
Standard Key	The system name of the rule. Approval policy steps bind to this key.	Example: XpCmdShellUsage
Starts with	The text searched for in some rules.	
Error Message	The explanation shown when the rule fires.	
Severity (Tier)	The risk level of the rule.	0=Info, 1=Low, 2=Medium, 3=High, 4=Critical
Never Skip	If it fires, automatic execution is disabled and the bound approval step cannot be skipped.	On / Off
Blocks Save	If it fires, the request cannot be saved at all.	On / Off
Blocked Query Types	In which request types the rule blocks saving.	
Max Value	The threshold for numeric rules.	
Database Type	Which databases the rule applies to. Semicolon separated. Empty means all.	1;3 = SQL Server and PostgreSQL

What happens on a violation

There are three possible outcomes:

Situation	Outcome
The rule fired, "Blocks Save" is off	The request is saved. The finding enters the risk calculation and may raise the band.
The rule fired, "Blocks Save" is on	The request is not saved. The user sees the error message and must fix the script.
The rule fired, "Never Skip" is on	The request is saved, but automatic execution is disabled and the related approval step is locked.

Example standards

Key	What it checks	Severity	Never skip
CriticalObjectDrop	Dropping a critical object.	Critical	Yes
XpCmdShellUsage	An xp_cmdshell call. Runs operating system commands.	Critical	Yes
LinkedServerUsage	External data source access (linked server, OPENROWSET, database link).	High	Yes
DynamicSqlUsage	Dynamic SQL. Carries injection risk and the executed query is not captured in the audit trail.	High	No
TempTableUsage	Temporary table usage.	Medium	No
WaitForDelayUsage	WAITFOR DELAY. Blocks the connection.	Medium	No
DeprecatedTypeUsage	Deprecated data types.	Medium	No
IdentityFunctionUsage	@@IDENTITY. May return the wrong value through trigger chains.	Medium	No
RequireTryCatch	No error handling block in a procedure body.	Low	No
PrintStatementInObject	A PRINT statement in an object body.	Low	No
OrderByPosition	Positional ordering such as ORDER BY 1, 2.	Info	No

The full list is on the Settings > SQL Standards screen.

9.3 Sensitive Data Patterns

Screen: Settings > Sensitive Data Patterns

Permission: settings.patterns

What it is for

Rules that look for sensitive data inside the values of a query result. They inspect the value itself, not the column name.

Fields

Field	What it does	Example
Name	Name of the pattern.	National ID
Regex	The format searched for.	An 11-digit number
Mask Style	Full or partial masking.	PartialMask
Keep Right	How many characters stay visible in partial masking.	4
Validator	The algorithm run after a format match.	TcknChecksum, Luhn, IbanMod97
Applies to numeric columns	Should numeric and date columns also be converted to text and scanned.	Off
Category	Sector or topic label.	Finance, Health
Built-in	Whether it ships with the product.	
Enabled	Should the pattern run.	

Why validators exist

Looking at the format alone produces false positives. Not every 11-digit number is an identity number; an order number can be 11 digits too.

Therefore a pattern match is not sufficient on its own. If a validator is defined, the algorithm must pass as well:

Validator	What it does
TcknChecksum	Verifies the check digit calculation of a national identity number.

Validator	What it does
Luhn	The check algorithm used for credit card numbers.
IbanMod97	Performs the modulo 97 check of an IBAN.

The result: an 11-digit order number cannot pass the identity check and is not masked. This reduces both unnecessary masking and requests to "open this column".

Partial masking example

With PartialMask and 4 characters kept on the right, the value 12345678901 is shown as *8901. Full masking replaces the value entirely with asterisks.

9.4 Sensitive Column Catalog

Screen: Settings > Sensitive Columns

Permission: settings.qrsensitive

What it is for

A classification catalog that works on the column name. Data values are never inspected, so nobody needs to see raw data in order to classify it.

Fields

Field	What it does	Values
Column Name	The name searched for.	
Match Mode	Exact, Contains or Regex.	
Schema Name	Limits the rule to this schema. Empty means every schema.	
Table Name	Limits the rule to this table. Empty means every table.	
Database Type	Limits the rule to this type. Empty means all.	
Classification	Auditor label.	KVKK-Identity, PCI-Card
Mask Style	Full or partial. Empty means full masking.	
Keep Right	For partial masking.	
Active	Should the rule run.	

The same column name can be defined in different scopes. Uniqueness is enforced together with the scope.

If a server's masking regime is "Content only", this catalog is disabled for that server and only patterns run.

9.5 Candidate Columns (the learning loop)

Screen: Settings > Candidate Columns

Permission: settings.qrsensitive

How a candidate appears

If a column caught by a pattern in a query result is not in the catalog, a candidate record is created, or the hit count of an existing candidate is increased.

No data value is ever stored in this table. Only the column name, table name, schema name, server name and the matched pattern name are kept.

Columns

Column	**What it shows**
Column Name	The column name in the result.
Source Schema / Table / Column	The real source of the column. If an alias was used, the real name appears here.
Server	Where it was seen.
Detected Pattern	Which pattern matched.
Hit Count	In how many deliveries it was encountered. Frequent ones are listed at the top.
First / Last Seen	Dates.
Status	New awaiting decision, Accepted added to the catalog, Ignored dismissed.
Decided by / Decided at	

What happens on acceptance

An accepted candidate is added to the sensitive column catalog. From then on the column is also caught by name, so it is masked even in rows whose values do not match the pattern.

Thanks to this loop the catalog grows richer with use, and the organisation does not have to scan its own data structures by hand.

Section 10. Approval Policies

Screen: Settings > Approval Policies

Permission: settings.approvals

10.1 What a policy is

A policy holds the answer to "whose approval is required in which situation". A policy contains ordered steps and each step asks for a role.

Multiple policies exist because different environments require different strictness. You may want three approvals in production and none in test.

10.2 Policy fields

Field	What it does	If left empty
Name	Name of the policy. Written to the request as a permanent trace.	Mandatory
Description	Free text.	-
Active	Is the policy in use.	An inactive policy never matches.
Emergency Policy	Is this policy only for emergency requests.	If off, only normal requests match.
Requester can self approve	Overrides the organisation default within this policy scope.	Empty means the organisation parameter is used.
Approver can execute	Same.	Empty means true.
Distinct approver per step	Same.	Empty means false.

Important: an emergency request matches only an emergency policy, and a normal request only a normal policy. The two sets never mix.

10.3 Policy mapping (scope)

Policies are not bound to requests directly. A mapping record sits in between and defines the scope:

Field	Meaning
Server	A specific server. Empty means not server-scoped.
Environment	Production, Staging, Test, Development. Empty means not environment-scoped.
Query Type	Change, QueryResult, Rollback. Empty means any type.

Field	Meaning
Priority	Which policy is tried first when several exist in the same scope. A higher number comes first.

What Priority does

If several policies are defined at the same scope layer, the one with the higher priority is tried first. On a tie the policy with the lower id comes first, which makes the outcome deterministic.

Priority is only compared within the same layer. Between different layers specificity, not priority, decides.

10.4 Policy selection flow

This flow runs when the approval steps are created after a request is saved. It runs separately for each server on the request.

```
For each server on the request:
  |
  v
1. Is there a mapping SPECIFIC to this server and to exactly this request type?
       yes -> these are the candidates, order by priority
       no  -> continue
  v
2. Is there a mapping SPECIFIC to this server with no type set?
       yes -> these are the candidates
       no  -> continue
  v
3. Is there a mapping for the server ENVIRONMENT and exactly this request type?
       yes -> these are the candidates
       no  -> continue
  v
4. Is there a mapping for the server ENVIRONMENT with no type set?
       yes -> these are the candidates
       no  -> continue
  v
5. Is there a GLOBAL mapping (no server, no environment) for exactly this type?
```

```
        yes -> these are the candidates
        no  -> continue
   v
6. GLOBAL mapping with no type set
   |
   v
Candidates are tried in priority order.
The first one that produces at least one active step wins.
   |
   v
Comparison across servers:
   - The most critical ENVIRONMENT wins first
     (Production > Staging > Test > Development > undefined)
   - At the same environment level, the one with MORE active steps wins
```

What happens when several policies match

The order above selects a single layer. Candidates in that layer are tried by priority and the first candidate producing at least one active step wins.

The reason for this behaviour: a high-priority policy may have conditional steps and none of them may fire. If that policy won, the lower-priority policy that would actually have required approval would be silently disabled.

What happens when none matches

If no policy matches, no approval step is created and the request is approved automatically. This is written to the audit trail as the ChangeRequest.AutoApproved event and the applied policy field on the request stays empty.

The Control Exceptions report lists this under "no matching policy". It is one of the first places auditors look.

Installation recommendation: define at least one global policy so that no request is left without a policy.

What happens when every step is skipped

If a policy matched but every step was skipped because of the conditions, the request is still auto approved. In this case the skipped steps are recorded with their reasons. The request detail and the evidence dossier show "which step was skipped and why".

How to tell the two cases apart: if the applied policy name is filled, a policy matched but steps were skipped; if it is empty, no policy matched at all.

10.5 Step fields and behaviour

The following fields are defined for each policy step.

Field	What it does	How it behaves for a given value
Order (StepOrder)	The position of the step.	Steps are created in this order. The same order cannot be used twice in one policy.
Step Name	The name shown on screen.	Example: "Manager Approval".
Approver Role	The lowest role that can close the step.	Setting DbAdmin lets DbAdmin and above approve.
Band At Least (TriggerIfBandAtLeast)	The step activates if the risk band is at or above this value.	Setting 3 (High) makes the step run only on High and Critical requests.
Band At Most (SkipIfBandAtMost)	The step is skipped if the risk band is at or below this value.	Setting 1 (Low) skips the step on Info and Low requests.
Triggering Standards (TriggerIfStandardKeys)	The step activates if one of these standards fired. Semicolon separated.	XpCmdShellUsage;LinkedServerUsage makes the step run only when those findings exist.
Requester Role At Least (SkipIfRequesterRoleAtLeast)	The step is skipped if the requester's role is at or above this level.	Setting ChangeManager skips the step on requests opened by a change manager.
Locked Step (IsNonSkippable)	The step cannot be skipped by any rule.	When on, band, role and every other skipping rule is void.

How a step decides

The following checks run in order for each step:

```
1. Is the step LOCKED (IsNonSkippable)?          yes -> NOT SKIPPED, decision ends
2. Did a never-skip standard that triggers
   this step fire?                               yes -> NOT SKIPPED, decision ends
3. Is this a TRIGGER step
   (it has standard keys or a band threshold)?
      yes and nothing matched                    -> SKIPPED ("condition not met")
      yes and something matched                  -> rules 5 and 6 below DO NOT APPLY
4. Does the requester role exceed the threshold?  yes -> SKIPPED ("requester role")
5. Is the requester role equal to or above
   the approver role of the step?                 yes -> SKIPPED ("owner role")
```

```
 6. Is the band below the skip threshold?          yes -> SKIPPED ("low band")

 7. If none of the above                           -> STEP IS ACTIVE and awaits
approval
```

A point to watch: in rule 3, a trigger step whose condition matched cannot be skipped by rules 5 and 6. A step defined as "activate on high risk" is not skipped because the requester holds a senior role. Otherwise seniority would remove the control from the person doing the risky work.

Example policy

A production policy can be built like this:

Order	Step Name	Approver Role	Condition	Locked
1	Team Lead Approval	RequestManager	Skip if band at most Info	No
2	Database Administrator Approval	DbAdmin	Band at least Medium	No
3	Change Manager Approval	ChangeManager	Band at least High	No
4	Security Approval	SystemManager	If XpCmdShellUsage;LinkedServerUsage fired	Yes

In this policy:

- An Info band request asks for no approval and is auto approved.

- A Medium band request asks for steps 1 and 2.

- A High band request asks for steps 1, 2 and 3.

- A request containing xp_cmdshell additionally asks for step 4, and that step can never be skipped.

10.6 Segregation of duties rules

Three settings answer three different questions:

Setting	Question
RequesterCanSelfApproveExecute	May the requester approve and execute their own request?
ApproverCanExecute	May the approver execute the same request?
RequireDistinctApproverPerStep	May the same person close more than one step?

They are defined first as organisation parameters (Section 7.4) and can then be overridden per policy. A field left empty in the policy uses the organisation default.

The freezing rule: the effective values are written onto the request when the approval flow is initialised. Even if the policy or parameter changes later, the request is evaluated with the rule set it entered under. The evidence dossier shows the rule of that day.

The role-skip trace

When a step is skipped because of a role rule, the step record is not deleted; it is stored with a "skipped" status and its reason. The decision maker is recorded as "System". This makes the audit question "why was this step never asked" answerable.

10.7 Deadlock scenarios

Scenario	Result	Fix
A step asks for DbAdmin but no DbAdmin exists	The request waits forever.	Add a user with that role or lower the step role.
RequireDistinctApproverPerStep is on, there are two steps and only one authorised person	The second step cannot be closed.	Add a second authorised person.
RequesterCanSelfApproveExecute is off and the only authorised person both opens and approves	They cannot approve their own request.	Add another approver.
No emergency policy is defined	The emergency option cannot be used.	See 10.8.

10.8 Emergency availability

For the emergency option to appear on a request:

1. A matching emergency policy must exist for every selected server.

2. Those policies must contain at least one locked step.

The second condition is deliberate. An emergency shortens the approval path, it does not remove it. Without any locked step, declaring an emergency would disable the approval mechanism entirely, which would be a governance gap.

The user's role must also pass the EmergencyMinimumRole threshold to raise an emergency request.

Section 11. Change Request End to End

11.1 The state machine

Value	Status	Meaning	Terminal
1	Pending	Awaiting approval.	No
2	Approved	Approved, waiting to be executed.	No
3	Executed	Executed.	Yes
4	Rejected	Rejected.	Yes
5	Manual	Declared as applied manually.	Yes
6	NoLongerNeeded	No longer required.	Yes
7	Executing	Currently running.	No

What a terminal state means: the request has left the active flow. The worker does not pick it up, no new action can be taken on it and it cannot be edited.

Transition matrix

Current state	Possible action	New state
Pending	Approve (when the last step closes)	Approved
Pending	Reject	Rejected
Pending	Mark manual	Manual
Pending	No longer needed	NoLongerNeeded
Pending	Edit (owner only)	Pending
Approved	Execute	Executing, then Executed or Rejected
Approved	Schedule	Approved (schedule information added)
Approved	Cancel schedule	Approved
Approved	Reject	Rejected
Approved	Mark manual	Manual
Approved	No longer needed	NoLongerNeeded
Executing	Cancel execution	Approved

Current state	Possible action	New state
Executed	Mark done	Executed (information updated)
Rejected, Manual, NoLongerNeeded	None	-

When execution fails, the behaviour depends on RejectOnError: true gives Rejected, false returns the request to Approved.

11.2 The request creation screen

Screen: Requests > New Request

Permission: cr.add

Field	What it does	Required	Notes
Description	A plain-language summary of what the request does.	Yes	Appears in lists and reports.
Request Type	Change or QueryResult.	Yes	Rollback is not chosen manually, it is generated.
Servers	Target servers. Several can be selected.	Yes	See Section 6.6.
Ticket Number	ITSM ticket number.	Depends on ChangeTicketRequired	Format is validated with ChangeTicketRegex.
Query Timeout	How long the script may run.	-	Lower bound DefaultQueryTimeout, upper bound MaxQueryTimeout.
Scripts	One or more SQL texts.	Yes	Each script has a name and an order.
Emergency	Is this an emergency request.	-	The conditions in Section 10.8 must hold for it to appear.
Emergency Reason	A mandatory written reason when emergency is selected.	Conditional	
Emergency Ticket	The ticket number of the emergency record.	Conditional	
Requested Execution Date	When you want the request to run.	-	Used in automatic scheduling.
Delay (minutes)	How many minutes to wait after approval.	-	Used when no date is supplied.

Additional fields for query result requests are described in Section 13.

11.3 The script editor and parsing

When parsing runs

Parsing is triggered after a script is written and runs again when the request is saved. The result at save time is stored permanently.

What parsing produces

Output	Description
Object list	Every object the script touches: database, schema, name, type.
Operation type	CREATE, ALTER, DROP, INSERT, UPDATE, DELETE, GRANT and so on.
Finding list	The SQL standards that fired, with their severities.
Critical object flag	Whether the touched object is in the catalog.
Risk band	From the evaluation of all findings.

Parsing errors

If a script cannot be parsed (a syntax error or an unsupported construct), it appears in the finding list as a parse error. In that case:

- The request can still be saved.

- The object list may stay empty.

- Standard-key triggers on approval steps do not fire, because it is unknown which rule was triggered. In this situation using band-based steps is a more cautious design.

11.4 The request detail screen

Screen: Requests > (select a request)

Header fields

Field	What it shows
Request number	The id of the record.
Description	The request description.
Status	One of the states in Section 11.1.

Field	What it shows
Request type	Change, QueryResult or Rollback.
Risk band and score	Section 12.
Servers	The semicolon-separated target list.
Ticket number	The ITSM ticket.
Emergency	The flag and the reason if it is an emergency request.
Applied policy	The policy matched when the approval flow was built. Empty means no policy matched.
Created by / Created at	
Execution requested by / at	The person who pressed the button. SYSTEM for automatic scheduling.
Executed by / Executed at	The service or user that carried out the work.
Rows affected	The number of rows changed by the execution.

The scripts and objects table

This table contains the columns most often asked about.

Column	What it shows	How it is derived	Example
Script	Which script it belongs to.	The script name.	01_add_table.sql
Schema	The schema of the object.	From parsing.	dbo
Object Name	The object touched.	From parsing.	Customer
Object Type	The type of the object.	From parsing.	Table, Procedure, View
Operation	The operation applied to the object.	From parsing.	ALTER, DROP, INSERT
Statement	The related SQL fragment or line information.	From parsing, with line and column numbers.	line 12
Critical Object	Whether the object is in the critical object catalog.	Compared against the catalog when the request is parsed and the decision is stored. It does not change even if the catalog changes later.	Ticked / Empty

Column	What it shows	How it is derived	Example
Risky Operation	Whether the statement fired a SQL standard.	It counts as risky when the severity of the fired standard is Medium or above.	Ticked / Empty
Message	The explanation of the fired standard.	The error message in the standard definition.	"Dynamic SQL usage detected."
Severity	The risk level of the finding.	The Tier in the standard definition.	High

The difference between Critical Object and Risky Operation:

- Critical Object is about *what* was touched. The organisation declared that object important.

- Risky Operation is about *what was done*. The operation itself violated a rule.

If a request performs a dangerous operation on an ordinary table, Risky Operation is ticked and Critical Object is empty. If it performs a harmless SELECT on a critical table, the opposite. When both are ticked, the request deserves the highest attention.

The approval steps table

Column	What it shows
Order	The step order.
Step Name	The name from the policy.
Required Role	The lowest role that can close the step.
Status	Pending, Approved, Rejected, Skipped.
Decided by	The user who closed the step. "System" for skipped steps.
Decided at	
Comment / Reason	The approval comment or the skip reason.
Override	Ticked if the step was passed with emergency authority.
Assigned to	If the requester's direct manager meets the step role, the step is assigned to them.

Execution result

Field	What it shows
Execution log	Which block of which script ran, how many rows were affected, and in which block an error occurred.
Rows affected	The total.
Error message	If any.
Partial execution warning	On Oracle targets, a note that blocks executed before the error may not have been rolled back.

Audit / event list

Audit records belonging to the request are listed here: creation, editing, approval, rejection, scheduling, execution, cancellation.

11.5 Buttons and enablement conditions

Every button has two possible behaviours:

- Hidden: the user can never perform this action (no role permission, wrong request type, not the owner, module not licensed).

- Disabled (greyed out): the user could perform it, but not right now (wrong state, active package, own request, source not yet applied). Hovering shows the reason.

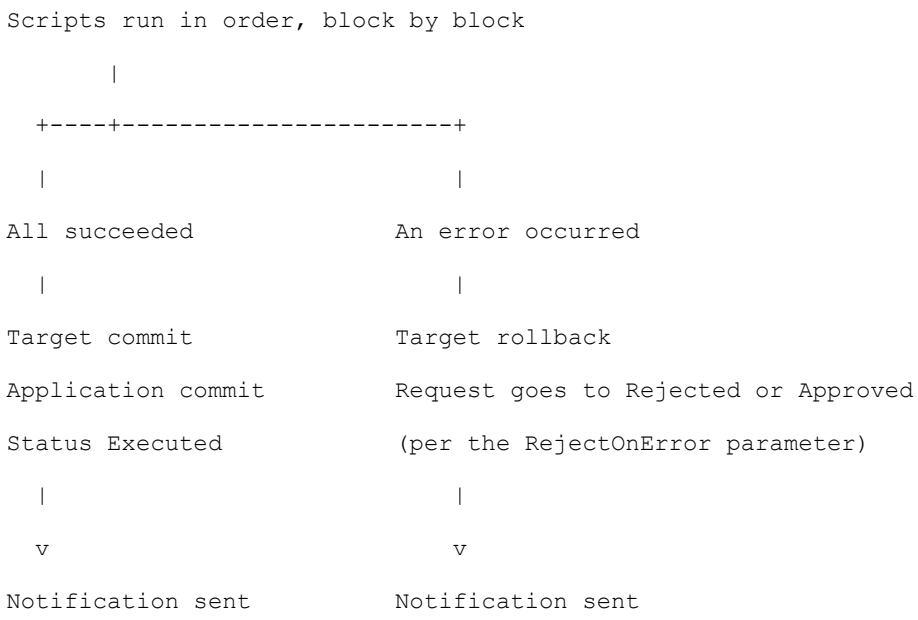
Button	Visible when	Enabled when
Edit	You are the owner	Not terminal, not in an active package
Approve	Approve permission	Status Pending, not in an active package, not your own request unless allowed, an eligible pending step exists
Reject	Approve permission	Status Pending or Approved
Execute	Execute permission; hidden for QR when QueryGovernance is off	Status Approved, not in an active package, rollback exists if required, segregation of duties rules do not block
Schedule	Approve permission and the scheduling module	Status Approved, same segregation of duties rules
Cancel schedule	Approve permission	Status Approved and a schedule exists
Mark done	Approve permission	Status Executed
Mark manual	Approve permission	Status Pending or Approved

Button	Visible when	Enabled when
No longer needed	Owner or approve permission	Status Pending or Approved
Generate rollback	Request type Change, owner, add permission, rollback module	Not terminal (requests that failed during execution are the exception), no active rollback exists
Copy	Add permission, request type is not Rollback	Always
Sandbox	Request type Change, sandbox permission and module	Not terminal
Cancel execution	Execute permission	Status Executing
Download result	QueryGovernance module, download rules	A result file exists

Button and backend consistency: the rule deciding whether a button is enabled and the rule deciding whether the backend accepts the request are the same code. This prevents the "the button was enabled but the action failed" situation.

11.6 The execution flow

```
The user presses "Execute"
       |
       v
Permission and state checks (CrPolicy)
       |
       v
The request moves to "Executing" and enters the queue
ExecutionRequestedBy = the person who pressed the button
       |
       v
The worker claims the queued request
       |
       v
A transaction is opened on the target server
       |
       v
```



Scheduled execution

In manual scheduling the user picks a date. It must be at least ExecutionDelaySeconds (and at least 30 seconds) in the future. When the time comes the worker picks the request up.

In automatic scheduling the date is computed by the system. See Section 6.4.

Cancelling an execution

A cancellation can be requested while a request is running. The system checks for a cancellation request every two seconds. When one arrives the running operation is stopped, the target transaction is rolled back and the request returns to Approved.

Crash recovery

If the worker service stops mid-run, the request stays in Executing. Such requests are detected against a time threshold and returned to Approved so that no request stays locked. The threshold is the Workers:StaleExecutionRecoveryMinutes setting and defaults to 120 minutes. For details and how to choose the value see Section 25.7.

11.7 Partial success and rollback

A request contains several blocks. If an error occurs:

Database	**Behaviour**
SQL Server, PostgreSQL	The transaction is rolled back. No change remains.
Oracle	DDL statements commit implicitly. DDL blocks executed before the error cannot be rolled back. This is written explicitly into the execution log.

For this reason a rollback script matters more on Oracle targets and may be made mandatory for requests containing DDL.

The error message states how many blocks ran and in which block execution stopped.

11.8 Rollback requests

- A rollback request is opened against a change request.

- Only the owner of the request can generate it.

- A request cannot have more than one active rollback at the same time.

- A rollback request goes through the normal approval flow. If the policy produces no step, the approval requirement of the source request is mirrored: the role authorised to approve the change also approves the reversal.

- A rollback can be executed only after the source request has been applied. The exception: if the source request failed during execution, a rollback can be run to clean up the partial change.

- Rollback requests are never scheduled automatically. A human must always execute them.

11.9 Execution attribution

When the worker executes, the "executed by" field names the service. That is not enough for audit purposes: the real question is who pressed the button.

A separate field is therefore kept:

Situation	**ExecutionRequestedBy**
The user executed manually	Username
The user scheduled manually	Username
The system scheduled automatically	SYSTEM
It ran inside a release package	The person who ran the package

If the schedule is cancelled this field is cleared. Otherwise a cancelled action would keep a record saying "this person started it".

11.10 Emergency override

On an emergency request, approval steps that are not locked can be passed by an authorised person. This action:

- Requires a written reason.

- Writes an "overridden" flag and the reason onto the step record.

- Is recorded as a separate audit event.

- Is listed in the Control Exceptions report.

- Is reflected in the exception counter on the dashboard.

Locked steps cannot be passed even in an emergency. An emergency shortens approval, it does not remove it.

The user's role must pass the EmergencyMinimumRole threshold to declare an emergency.

11.11 The request list screen

Column	What it shows
Number	Request id.
Description	Request description.
Type	Change, QueryResult, Rollback.
Status	Section 11.1.
Risk	Band and score.
Servers	Semicolon separated.
Ticket	The ITSM ticket.
Created by	
Created at	In your browser's local time.
Approval state	Which role is being awaited if a step is pending.
Schedule	The execution time if set.

Filters: status, type, server, date range, creator, ticket number, free text.

A user without the cr.seeall permission sees only their own requests.

Section 12. Risk Score

12.1 Purpose and principles

The risk model answers a single question: "how much attention does this request need?"

The model has three principles:

1. The worst case decides. There is no average, sum or multiplier. The highest finding sets the band.

2. Decisions use the band, not the number. Opening approval steps or disabling automatic execution look at the band. The numeric score exists for display only.

3. The result is deterministic. The same findings always produce the same band and the same score.

12.2 What a band is

The band is the risk level of the request. There are five steps:

Band	Numeric value	Meaning
Info	0	Informational. Not counted as risk.
Low	1	Low risk.
Medium	2	Medium risk.
High	3	High risk.
Critical	4	Critical risk.

Every SQL standard has a severity (Tier), and that severity is one of the five steps above.

12.3 How the band is determined

```
Scripts are parsed
       |
       v
The fired standard findings are collected
       |
       v
Findings sharing a standard key form ONE GROUP
(the severity of the group = the highest severity in it)
       |
       v
BAND = the HIGHEST severity across all groups
```

With no findings the band is Info.

A critical detail: grouping is by standard key. The same standard firing ten times inside a script counts as one group.

12.4 How the score is computed

The score is a number between 0 and 100 and a higher number means safer. Every band has its own range:

Band	Score range
Info	90 - 100
Low	70 - 89
Medium	50 - 69
High	25 - 49
Critical	0 - 24

The calculation is:

```
range_high = the upper bound of the band
range_low  = the lower bound of the band

top_tier_count = the number of DISTINCT standards at the band-setting severity

if top_tier_count <= 0
    score = range_high
else
    drop  = min(top_tier_count - 1, range_high - range_low)
    score = range_high - drop
```

In other words, the more distinct findings there are at the top severity, the lower the score sits inside the range, but it never leaves the band.

12.5 Worked example: band High with three High findings

A request fired the following:

Standard	Severity	Times
DynamicSqlUsage	High	2

Standard	Severity	Times
NoUpdateWithoutWhere	High	1
NoTruncateTable	High	5
TempTableUsage	Medium	3
PrintStatementInObject	Low	1

Step by step:

1. Grouping: there are five distinct standard keys, so five groups. NoTruncateTable fired five times but counts as one group.

2. Band: the highest severity among the groups is High. Band = High.

3. Top tier count: the number of groups at High severity = 3 (DynamicSqlUsage, NoUpdateWithoutWhere, NoTruncateTable).

4. Range: 25 - 49 for High.

5. Drop: min(3 - 1, 49 - 25) = min(2, 24) = 2.

6. Score: 49 - 2 = 47.

Result: band High, score 47.

Variations of the same example

Case	Top tier group count	Score
One High finding	1	49 - 0 = 49
Two distinct High findings	2	49 - 1 = 48
Three distinct High findings	3	49 - 2 = 47
The same High standard fired 10 times	1	49 - 0 = 49
30 distinct High findings	30	49 - min(29, 24) = 49 - 24 = 25
No findings at all	0	Band Info, score 100
One Critical finding	1	Band Critical, 24 - 0 = 24

The last row matters: a single Critical finding produces a lower (riskier) score than thirty High findings. That is deliberate; the worst case decides.

12.6 The score explanation (ScoreExplain)

Showing only a number is not enough. The system produces an explanation for every request:

Field	What it shows
Band	The computed risk band.
Score	The 0-100 number.
Determining finding	The finding that set the band. It is shown first in the list.
Finding list	For each finding: standard key, severity, message, how many times it fired, never-skip flag.
Effect	Whether each finding set the band. Determining set it, Neutral did not.
Has never-skip	Whether any finding is flagged never-skip.

Findings are sorted by severity descending and then alphabetically by key, so the most important finding is always at the top.

12.7 Never-skip rules

A SQL standard can be flagged never-skip. When such a standard fires:

1. Automatic execution does not happen. Even after approval the system does not run the request on its own. Any schedule previously set by the system is cleared.

2. The approval step bound to it is locked. Whatever the skip conditions are, the step runs.

3. The finding is additionally flagged in the request detail and in the evidence dossier.

Standards flagged never-skip by default include CriticalObjectDrop, XpCmdShellUsage, LinkedServerUsage and similar highest-risk rules. The full list is visible on the Settings > SQL Standards screen.

What happens if someone tries to skip it: there is no way to skip a locked step. Even an emergency override cannot pass it.

12.8 Where the score appears

Place	How it appears
Request list	Band badge and score.
Request detail	Band, score, finding list, explanation.
Dashboard	Risk distribution chart.

Place	How it appears
High Risk Changes report	Band and score columns.
Evidence dossier	Risk summary and finding list.

12.9 How the score affects decisions

Decision	What it looks at
Activating an approval step	The band (TriggerIfBandAtLeast).
Skipping an approval step	The band (SkipIfBandAtMost).
Automatic execution	The presence of a never-skip finding.
Whether a request can be saved	Standards flagged "blocks save".

The numeric score is used in no decision. It exists for human reading only.

Section 13. Query Result Flow

13.1 What it is for

Every organisation needs to read data from live environments: investigating a customer complaint, finding the root cause of a fault, validating a report. That need is usually met without controls; someone is told "run this query and send me the result".

The Query Result feature puts this work on the record:

- The query is opened as a request and goes through approval.

- The result is not displayed on screen; it is packaged as an encrypted file.

- Sensitive columns are masked automatically.

- Who took which data for which stated reason is recorded.

This feature depends on the QueryGovernance module.

13.2 Query types

Value	Type	Meaning
0	None	Undefined.
1	Change	A standard change request.
2	QueryResult	A data reading request. It cannot be rolled back and the result is delivered as a file.
3	Rollback	A rollback request bound to a change.

13.3 Creating a query result request

In addition to the fields in Section 11.2:

Field	What it does	Required
To	Email addresses that will receive the result. Semicolon separated.	No
CC	Copy recipients.	No
Requested unmasked columns	Columns requested without masking. Hidden when UnmaskedColumnRequestPolicy is Disabled.	No
Unmasking reason	A reason when the list above is filled.	Conditional

The script count rule: a query result request can carry only one script. If you need to run several queries, combine them into a single script; every result set appears as a separate Excel sheet in the output file. The sheet name is built from the server name and a sequence number, for example PROD-SQL01_01.

Column preview

While the request is saved, if QrColumnPreviewEnabled is on, the system connects to the target server and asks which columns the query will return.

This operation does not run the query. Only schema information is requested, no row is read and no side effect occurs on the target.

Why it matters: the preview learns the real source of aliased columns.

```
SELECT TCKN AS CustomerCode FROM Customer
```

The returned column is named CustomerCode. A catalog that looks at column names cannot catch it. The preview learns that the source column is TCKN, and that information is re-attached at execution time. Masking therefore cannot be bypassed with an alias.

The preview result is stored permanently. The approver sees the same list the requester saw, and the evidence dossier allows "expected columns" to be compared with "actual masking".

If the server cannot answer, the request is still saved; the preview is not mandatory.

The email approval step

If QrEmailApprovalPolicy is RequireApproval, the addresses in the recipient list are checked. If a previously unapproved address is present, an extra approval step is added and closed by a user holding the QrEmailApproverRole role.

If QrEmailAutoApproveUsers is True, addresses belonging to registered users count as approved.

The unmasked column approval step

According to UnmaskedColumnRequestPolicy:

Policy	**Behaviour**
RequireApproval	An extra approval step is added. Someone holding UnmaskedColumnApproverRole approves it.
AllowWithReason	No extra step; the requester counts as auto-approved on their own behalf.
Disabled	The feature is off. Any stored list is cleared.

If QrMaskingEnabled is off, the policy behaves as Disabled.

13.4 What happens behind the scenes at execution

This is the flow people ask about most. Step by step:

```
 1. The request is executed (manually or on schedule)
 2. ALL target servers of the request are visited in turn
 3. On each server:
      a. A connection is opened and a transaction is started
      b. Scripts run block by block and result sets are read
      c. Column metadata is attached
      d. The COLUMN PREVIEW taken at save time is re-attached
         (the source column of an alias comes from here)
      e. Rows are loaded into memory
 4. After data is collected from all servers:
      the MASKING DECISION is made
 5. The decision is made SEPARATELY for each server
      (production can be masked while test stays open)
 6. The masking record is written (one row per server)
      The record is written EVEN IF the policy is off
 7. The results are turned into an Excel file
 8. The file is placed in an encrypted ZIP and a password is generated
 9. The password is encrypted and written onto the request record
10. If there are email recipients, delivery is queued
11. The execution log is written:
      how many rows returned, which columns were masked,
      which columns went out open, whether the time budget was exceeded
12. The request moves to Executed
```

How the masking decision is made

```
 Is QrMaskingEnabled off?
      yes -> No detection and no masking is performed.
             The record is STILL written with policy = MaskingOff.
             This is the only place an auditor can see this situation.
      no  -> continue
```

```
        |
        v
Is QrSensitiveDataPolicy = WarnOnly?
     yes -> Detection runs, nothing is masked.
            Found columns are recorded as "sent open".
     no  -> AutoMask (an unknown value also counts as AutoMask)
        |
        v
For each server, for each column:
        |
        v
Is the server regime "Content only"?
     yes -> the column name catalog is SKIPPED, only pattern checks run
     no  -> both catalog and pattern checks run
        |
        v
Is the column sensitive?
     no  -> untouched
     yes -> continue
        |
        v
Did the user request this column unmasked and was it APPROVED?
     yes -> left open, reason recorded as "UnmaskApproved"
     no  -> MASKED, reason recorded as "Catalog" or "Pattern"
```

If the time budget is exceeded, unscanned columns are marked "not scanned" and masked. The safe side always wins.

What is recorded

Two tables are written:

Summary record (one row per server): request number, server name, delivery time, sender, masked columns, unmasked columns, matched patterns, applied policy, the server regime at delivery time, exemption reason, detection settings, whether an exception was approved.

Detail record (one row per column): column name, source column name, decision (masked / not masked), reason type, reason detail, applied mask style.

Reason types:

Type	Meaning
Catalog	The column name matched the sensitive column catalog.
Pattern	The column values matched a sensitive data pattern.
UnmaskApproved	The user requested it, it was approved and left open.
NotScanned	Not scanned because of the time budget, masked on the safe side.
PolicyWarnOnly	Not masked because the policy is WarnOnly.
MaskingOff	No detection was performed because the master switch was off.

Freezing the detection settings matters. If the threshold or the sample size is changed later, the question "what was the threshold on this delivery" cannot be answered anywhere else.

13.5 The result file and delivery

Topic	Behaviour
File format	An Excel file inside an encrypted ZIP.
Password	Generated separately for every delivery.
Where the password lives	Stored encrypted on the request record and shown to authorised users.
Email	Sent if there are recipients. Files exceeding MaxMailAttachmentMB are not attached.
Download	From the request detail.
Retention	QrFileRetentionDays days. When it expires the maintenance task deletes the file and the deletion is recorded.

Download permission

```
Is the person downloading the owner of the request?
     yes -> Is QrRequesterCanDownload true?
               yes -> can download
               no  -> cannot download
     no  -> Does their role pass the QrDownloadMinimumRole threshold?
```

```
yes -> can download

no  -> cannot download
```

Every download is written to the audit trail and appears in the Data Access and Export Log report.

13.6 Special rules in this flow

Rule	Reason
Query result requests cannot run inside a release package.	The package flow produces no masking and no record; unmasked data would leave without a trace.
The target transaction is never committed for query result requests.	It is a read operation and leaves no change on the target.
If one server fails, the whole delivery is treated as failed.	Delivering a partial result would be misleading.
Writing a candidate column never blocks delivery.	A failure of the learning loop must not stop data delivery; the error is logged.

Section 14. Sandbox

14.1 What it is for

Sandbox means trying a change request on a real database without making it permanent. The script runs inside a transaction, the outcome is recorded, and the transaction is rolled back.

The aim is to see whether the script actually runs before it reaches production. Syntax errors, missing objects and permission problems are caught here.

This feature depends on the Sandbox module.

14.2 How it works

```
The user presses "Sandbox" and picks a server
        |
        v
A sandbox record is created (in processing state)
        |
        v
The worker picks up pending sandbox records
        |
        v
A transaction is opened on the target
        |
        v
Scripts run block by block and a log is collected
        |
        v
The transaction is ALWAYS rolled back
        |
        v
The outcome is recorded: success / failure plus the log
```

14.3 Effect on the flow

Outcome	Effect
Sandbox succeeded	The request continues on its normal path. It appears as evidence in the Sandbox Test Results report.

Outcome	Effect
Sandbox failed	The request is not blocked, but the result is visible in the detail and in the report. The approver is expected to see it.
The request was edited after the sandbox	The sandbox result is marked outdated. An outdated test does not cover the current scripts.

14.4 Known limits

Requests containing DDL cannot be sandboxed on Oracle targets. The reason: on Oracle, statements such as CREATE, ALTER, DROP, GRANT, REVOKE and TRUNCATE commit implicitly. The rollback at the end of the sandbox does not undo them, so the "trial" would leave a permanent change.

In that case the sandbox attempt is blocked and the statement types responsible are listed in the message.

Only Oracle scripts that manipulate data (INSERT, UPDATE, DELETE, MERGE, SELECT) can be sandboxed safely.

On SQL Server and PostgreSQL targets everything including DDL can be rolled back.

Section 15. Release Package

15.1 What it is for

It makes several interdependent requests run together in the right order. Example: the table must be added first, then the procedure updated, then the data migrated.

This feature depends on the ReleasePackages module.

15.2 Package fields

Field	What it does
Name	Name of the package.
Description	Free text.
Server	The server the package runs on.
Execution Mode	See 15.3.
Schedule	The date the package will run.
Query Timeout	The upper bound is the MaxQueryTimeout parameter.
Status	Preparing, queued, executed.
Success	Whether the execution succeeded.

15.3 Execution modes

Value	Mode	Behaviour
1	AllOrNothing	Every request runs inside a single transaction. If one fails, all are rolled back.
2	StopOnError	Each request runs in its own transaction. On an error the package stops and the successful ones so far remain.
3	ContinueOnError	Each request runs in its own transaction. On an error that request and its dependants are skipped and the rest continue.

Which mode to choose:

- Use AllOrNothing when the changes form an inseparable whole.

- Use StopOnError when order matters and you want to stop and investigate on the first error.

- Use ContinueOnError when the requests are largely independent and one failure should not stop the whole release.

15.4 Package content and ordering

Field	What it does
Sort Order	The order in which requests run.
Depends On Request	Which request must succeed first. In ContinueOnError mode the skip chain follows this field.
Executed	Whether the item ran.
Execution Description	The item-level outcome.

15.5 Differences from the single request flow

Topic	Single request	Release package
Transaction	One per request.	One or per request depending on the mode.
Approval	Per request.	The requests must already be approved individually.
Actions on the request	Free.	Once a request is in an active package, the edit, approve, execute and schedule buttons are disabled. The request is now governed by the package.
Automatic scheduling	Possible.	Not performed for a request in an active package.

15.6 Constraints

- Query result requests cannot be added to a package. Masking and delivery records are produced only in their own flow.

- A request can belong to only one active package at a time.

- Package execution time is also driven by MainWorkJob; due packages run first, then individual requests are processed.

Section 16. Object History (DDL History)

16.1 What it is for

It shows when and how objects on target servers changed. The data is read from the target database itself, not from SQL Change Guard records.

This feature depends on the InsightsReporting module.

16.2 Screen flow

```
Pick a server
    |
    v
Pick a database
    |
    v
Search / list objects
    |
    v
Pick an object -> view its history
```

Step	What it shows
Server list	Defined and active servers.
Database list	Databases on the selected server.
Object summary	Object name, schema, type, last change date. Filtered with the search box.
Object history	Versions of the selected object and their change times.

16.3 Audit value

The real benefit of this screen is making out-of-band changes visible.

If an object was changed through SQL Change Guard, there is a matching request record. If object history shows a change on a date with no request at all, that change was made outside the system.

The question to ask in an audit: "who made this change, under whose authority, without a record?"

The Object Change Frequency and Configuration Changes reports are also used when making this comparison.

Section 17. Audit Trail and Immutability

Screen: Audit Log

Permission: audit.view

17.1 What an audit record holds

Every audit row carries these fields:

Field	What it shows
Action	What happened. Example: ChangeRequest.Approved.
Entity Type	Which kind of record was affected. Example: ChangeRequest, Server, SettingChange.
Entity Id	The number of the affected record.
Old Value	The state before the change.
New Value	The state after the change.
Performed By	Username or system actor.
IP Address	The address the request came from.
Occurred At	The event time, stored in UTC.
Hash	The signature of the record.
Previous Hash	The signature of the previous record.
Hash Version	1, 2 or 3. See Section 5.4.
Key Id	Which key was used on version 3 records.

17.2 What is not held

- Query result data never enters the audit trail.

- Passwords and secrets are written masked.

- Column contents and row values are not stored.

17.3 Event naming convention

Event names follow the EntityType.Action form:

Example event	Meaning
ChangeRequest.Create	A request was created.
ChangeRequest.Approve	A request was approved.

Example event	Meaning
ChangeRequest.AutoApproved	The system approved automatically because no approval step was produced.
ChangeRequest.Reject	A request was rejected.
ChangeRequest.Executed	A request was executed.
SettingChange.Requested	A settings change was sent for approval.
SettingChange.Approved	A settings change was approved and applied.
SettingChange.Rejected	A settings change was rejected.
SettingChange.ApplyFailed	It was approved but could not be applied.
User.SessionsRevoked	The session renewal records of a user were revoked.
Role.SessionsRevoked	The session renewal records of everyone holding a role were revoked.
EvidencePackageExported	An evidence dossier was exported.

Control weakening events are recorded under separate names. An ordinary settings update and the loosening of a control are not written under the same event name. An auditor can search for the moment a control was weakened using a single event name.

17.4 The old and new value contract

Two rules apply:

1. Old value and new value are written in the same shape. In different shapes they could not be compared side by side.

2. The value is read back from the database after the write. What lands in the record is therefore the reality in the database, not the application's intention.

Nothing is applied for rejected settings changes. In that case the requested values are stored in the audit row; no other trace exists.

17.5 The rule set snapshot ("what applied that day")

When a request is created, the rule set in force at that moment is frozen onto the request:

- The id and the name of the applied approval policy.

- The effective segregation of duties settings.

- The critical object decisions.

- The fired standards and their severities.

Even if the policy is later deleted or changed, the audit trail of the request is not broken. An auditor can answer "which rules evaluated this request" from a single record.

17.6 The hash chain and verification

How the chain works is described in Section 5.4. This section covers the verification tools.

Verify Chain

This action on the Audit Log screen checks the whole chain from start to finish:

Check	What it looks at
Record consistency	Does the digest of each record match its own content.
Chain linkage	Is the "previous digest" field really the digest of the preceding record.
Key availability	Is the key of each version 3 record present in the ring.

The result lists the ids of any broken records.

Verify Immutable

This action compares the records in the main database with the sealed copy in the separate database.

Why it is needed: an attacker who obtains the HMAC key could rewrite the whole chain in the main database and "Verify Chain" would come back clean. The copy in the separate database is outside their reach.

The results returned:

Field	Meaning
Enabled	Is the immutable layer switched on.
Total audit records	The number of rows in the main database.
Protected from	The first record id from which copying started. Records before this id are unprotected.
Protected count	The number of records inside the protection scope.
Unprotected count	The number of records from before protection started.
Matched	Records identical to their copy.
Mismatched	Records whose digest differs from the copy.
Missing copy	Present in the main database, absent in the copy.
Orphan copy	Present in the copy, absent in the main database.

How to read the results:

Finding	Meaning	Urgency
Mismatched is zero	Cryptographic integrity is intact.	-
Mismatched is greater than zero	Evidence of manipulation. The digest of an append-only audit row never legitimately changes.	High
Missing copy	May have a legitimate cause: the second database was unreachable at that moment and the copy could not be written. Investigated as an anomaly.	Medium
Orphan copy	May have a legitimate cause: the transaction that wrote the audit record was rolled back while the copy, committed on a separate connection, remained.	Low

Missing and orphan records do not affect the integrity flag, because they would raise false alarms. Only a mismatched record is sufficient evidence on its own.

Current limitation: the immutable verifier is implemented for SQL Server only. If the application database runs on PostgreSQL or Oracle, this action returns a "disabled" result. This is a deliberate status report, not an error.

Verify Script Integrity

Checks that the scripts of a specific request have not changed since they were saved. Run from the request detail.

17.7 The audit integrity job

AuditIntegrityJob runs inside the worker at regular intervals and checks the chain. If it finds corruption it raises a critical notification, so manipulation is noticed without waiting for a human to open a screen.

17.8 The Audit Log screen

Column	What it shows
Time	Event time, in your browser's local time.
Action	The event name.

Column	What it shows
Entity Type	The affected record type.
Entity Id	The affected record number.
Performed By	The user or system actor.
IP	The request address.
Old Value / New Value	Shown side by side when the row is expanded.
Verification	Whether the signature of the record is valid.

Filters: date range, event name, entity type, entity id, user.

Section 18. Evidence Dossier

18.1 What it is for

It collects the full lifecycle evidence of a single request into one ZIP file in a form that can be handed to an auditor.

Usage: the auditor says "show me the evidence for that change". The file is exported from the request detail and given to them. The auditor can review and verify it without entering the system.

18.2 Who produces it

Two layers of permission are required for export:

1. The audit.evidenceexport permission. It comes from the role definition. By default it is on for the Auditor, DbAdmin, ChangeManager and SystemManager roles.

2. Permission to see the request. Without cr.seeall, a user can only export dossiers for their own requests.

The reason for two layers: an export permission must not grant access to the data of a request you cannot see.

18.3 File content

File	Content
Dossier.json	Canonical data. The request header, approval steps, scripts, risk summary, sandbox results and audit trail. The package digest is computed over this file.
Manifest.json	The package digest (packageHash) and the audit record id. Verification uses these two values.
Dossier.pdf	The same information in readable, printable form.

The audit trail is limited to 500 records; for requests with very long histories the most relevant records are included.

18.4 Why query result data is not included

Query result data is never placed into the evidence dossier.

The reason: the dossier exists to prove that the control mechanism worked, not to carry the data itself. Placing result data in the file would open a side door that bypasses masking and download permission checks entirely. Someone allowed to download a dossier would reach live data without holding the result file permission.

What the file does contain: the query text, how many rows returned, which columns were masked, which columns went out open and why.

What it does not contain: the row contents themselves.

18.5 The export itself is audited

When the file is produced, the EvidencePackageExported event is written to the audit trail. This record:

- Holds who produced a dossier, when, and for which request.

- Carries the package digest (packageHash).

- Appears in the Data Access and Export Log report.

In other words, the act of "producing evidence" is itself evidenced.

18.6 How the file is verified

1. The auditor opens the ZIP file they hold
2. They read two values from Manifest.json:
    - packageHash
    - auditEntryId
3. They open the "Verify Evidence" action on the Audit Log screen
4. They enter the two values
5. The system finds the audit record with that id
6. It compares the packageHash inside the record with the value entered

The result returned:

Field	**Meaning**
Is valid	Are the two digests identical.
Audit record id	The record compared.
Exported at	When the file was produced.
Exported by	The user who produced it.
Expected hash	The value in the audit record.
Provided hash	The value the auditor entered.

What a failed verification means

Situation	Meaning
Record not found	There is no export record with that id. The file may not have come from this system, or the id was entered incorrectly.
Digests differ	The file was altered after it was produced. Its content cannot be trusted.
Digests match	The file is exactly as it was when produced.

Verification proves that the file has not changed, not that its content is correct. The correctness of the content comes from the integrity of the audit chain (Section 17.6).

18.7 What an auditor can prove with this file

Question	The answer in the file
Who asked for this change?	The request header.
What did it change?	The scripts and the parsed object list.
What was its risk?	The risk band, score and finding list.
Under which rules was it evaluated?	The applied policy name and the frozen segregation of duties settings.
Who approved it?	The approval steps, decision makers and times.
Were any steps skipped, and why?	The skipped steps with their reasons.
Was it tested?	The sandbox results.
When and by whom was it executed?	The person who started the execution, the executing service, the times and the outcome.
Has this document been altered?	The package digest verification.

Section 19. Dashboard

Screen: Dashboard

Permission: dashboard.view

19.1 The time separation

Numbers on the panel fall into two groups and are shown separately:

Group	Meaning
Range	The number of things that happened inside the selected date range. It looks backwards.
Now	The number of things currently waiting. It is independent of the date range.

This separation matters. Mixing the two into one row of cards puts unrelated numbers such as "total 500, pending 12" side by side and the panel is read incorrectly.

19.2 Range cards

Card	What it shows	How it is computed
Created	Requests opened in the range.	By creation date.
Executed	Requests successfully executed in the range.	By execution date.
Failed	Requests whose execution attempt failed in the range.	
Rejected	Requests rejected in the range.	

19.3 Now cards

Card	What it shows
Awaiting approval	Requests currently in Pending.
Awaiting execution	Approved requests not yet executed.
Running	Requests currently in Executing.
Scheduled in the next 24 hours	Requests scheduled between now and tomorrow.
Pending settings approvals	Settings changes awaiting a decision.

19.4 Execution success rate

The formula is successful / (successful + failed).

Rejected requests are excluded, because a rejected request was never executed and does not count as a failure.

If no execution was attempted in the range, the card is blank. Writing zero would read as "everything failed".

19.5 Charts and lists

Section	What it shows	How to read it
Trend	The number of requests created and executed over time.	The gap between the two lines shows accumulated backlog. If creation stays above execution, the queue is growing.
Top users	The users who opened the most requests in the range.	Heavy concentration on one person can point to a concentration of knowledge and authority.
Server load	Requests per server.	Shows which environment receives the most change.
Longest waiting	The oldest pending requests.	Request number, description, ticket, opener, opening time, status, and which step and role it waits on.
Risk distribution	Request counts and percentages by band.	Records with no band (not calculated) appear separately. A growing Critical slice needs attention.
Approval queue	Pending step counts and average waiting time by role.	Shows which role is the bottleneck. A rising average wait indicates that role does not have enough people.

19.6 The exceptions card

The number of requests that left the normal flow. It is the dashboard answer to the auditor and management question "is the control working".

Counter	What it counts
Emergency	Emergency requests opened in the range.
Overridden steps	Approval steps passed with emergency authority.
Self approved	Cases where the requester approved their own request.

Counter	What it counts
No policy matched	Requests where no approval policy matched.
Unapproved unmasking	Requests where unmasked columns were requested but not approved.

If any of these is higher than expected, the detail is in the Control Exceptions report.

19.7 Differences by role

The panel narrows according to the user's permissions. A user without cr.seeall sees numbers only for their own requests. The settings approval counter is hidden for users without the relevant permission.

19.8 Empty data

If there are no records in the selected range, the cards show zero and the charts show an empty-state message. This is not an error.

Section 20. Reports

Screen: Reports

Permission: report.view

Module: InsightsReporting

20.1 General rules

- Reports are organised into five groups and there are 30 reports in total.

- Report output is produced in English. The reason is that reports are used in audit and compliance work, may be shared outside the organisation, and should be readable by international auditors.

- Every report has its own parameters. Date range parameters default to the last 30 days.

- Some reports are role restricted. The restriction is applied on the backend both in the list and at execution.

- Report queries are subject to a timeout (Reports:QueryTimeoutSeconds, default 500 seconds).

20.2 Common parameters

Parameter	Type	Meaning
Start Date	Date	Start of the range. Default: 30 days ago.
End Date	Date	End of the range. Default: today.
Server	List	Server filter. On multi-server requests the search runs inside the list.
Status	List	Request status.
Object Name	Text	Object name filter.
ITSM Ticket No	Text	Ticket number.
Request No	Number	Request number.
Minimum Risk Band	List	Risk band threshold. Default: 3 (High).
Statement Category	List	DDL, DML, DCL and so on.
Statement Type	List	CREATE, ALTER, DROP and so on.
Environment	List	Environment.
Database Type	List	Database type.
Principal / Account	Text	Database account.
User	Text	Application user.
Record Type	Text	Audit record type.

Parameter	Type	Meaning
Include deleted accounts	Checkbox	Should deleted accounts be included.

Date filters work on local day boundaries. See Section 23.5.

20.3 Change Governance group

Report	What it shows	The question it answers	Restriction
Change Request Summary	One line per request: status, risk band, applied approval policy, emergency flag, who started the execution, rows affected.	"Which changes were made in this period?"	-
Change Request by Object	Which request touched which database object.	"Who changed that table in the last six months?"	-
Work in Progress	Requests not yet executed and how many days they have waited.	"What is in the queue and for how long?"	-
Rejected and Cancelled Requests	Rejected or cancelled requests with the decision maker and the stated reason. The decision maker is read from the approval step, so later automatic updates cannot overwrite it.	"Why was it rejected and by whom?"	-
High Risk Changes	Requests at or above the selected band, with the emergency flag and applied policy. The score is shown as a secondary value.	"Which were the riskiest changes?"	-
Rollback Changes	Rollback requests together with their source requests.	"Which changes were reversed?"	-
Approval and Execution Duration	Elapsed hours between request, last approval and execution.	"Where is the process getting stuck?"	-
Execution Activity	What actually ran on the target servers: who started it, which service executed it, how long it took, rows affected, outcome and error text. For an automatic schedule the starter shows	"What ran in production?"	-

Report	What it shows	The question it answers	Restriction
	as SYSTEM. If it ran inside a release package, the package is shown too.		
Approval Step Detail	Every approval step: required role, decision maker, decision time, override flag and reason.	"Was dual control really applied?"	-
Sandbox Test Results	Test server, outcome, who ran it and whether the request was edited after the test. An outdated test does not cover the current scripts.	"Was it tried before going live?"	-
Control Exceptions	Requests that left the normal flow: emergency records, requests with no matching policy, overridden steps, self approved steps and unmasked data requests without approval.	"Where were the controls bypassed?"	Auditor, DbAdmin, ChangeManager, SystemManager

Auditors usually start with the Control Exceptions report. If it is empty, the control mechanism worked without exception.

20.4 Object & Structure group

Report	What it shows	How to read it
Critical Object Access	Requests touching an object marked critical. Criticality is the decision recorded when the request was parsed, so later catalog edits do not rewrite history.	Every row here deserves individual review.
Object Change Frequency	How often procedures, functions, triggers, views and tables change. The most frequently changed appear first.	Objects at the top signal architectural debt or instability.
Table Structure Changes	ALTER TABLE ADD and DROP COLUMN operations with the affected column names and counts.	The operations with the highest data loss risk are here.
Data Modification Frequency	How often data is modified per table through INSERT, UPDATE, DELETE,	Tables receiving frequent manual data fixes point to a process problem.

Report	What it shows	How to read it
	MERGE and procedure calls, with the last execution date.	
Multi Release Objects	Objects changed under more than one ticket.	Shows hot spots where several teams keep editing the same object.
Active Conflicts	Open requests about to change the same object in the same environment under different tickets.	Used to catch collisions before they reach production.

20.5 Access & Authorization group

Report	What it shows	Restriction
Privileged Account Operations	All authorisation statements inside requests: GRANT, REVOKE, DENY, account and role management, with the target principal and the privileges involved.	Auditor, DbAdmin, ChangeManager, SystemManager
User DDL Activity	Per user counts of CREATE, ALTER, DROP and authorisation operations, plus how many of that user's requests were high risk or emergency.	-
User Permission Matrix	Every account with its role, hierarchy level, account status, last login and the full permission set of that role. Access control evidence in one table.	Auditor, DbAdmin, ChangeManager, SystemManager
User Access Changes	The lifecycle of accounts and roles: who created an account, who changed permissions, who deleted, reactivated or unlocked it, and from which address.	Auditor, DbAdmin, ChangeManager, SystemManager

20.6 Data & Privacy group

Report	What it shows	Restriction
Query Result Activity	Every request that read production data: returned row count, recipient address, requested unmasked columns and the masking decision applied on delivery.	Auditor, DbAdmin, ChangeManager, SystemManager

Report	What it shows	Restriction
Sensitive Column Touches	Structural changes that add or drop a column registered in the sensitive data catalog, with the classification and masking style of that column. Exact and contains rules are applied the same way the masking engine applies them, and a rule limited to a schema or table only matches inside that scope. Regex rules are matched by exact column name here.	-
Masking Activity	What masking actually did on each delivery: the applied policy and regime, how many columns were masked, how many were delivered unmasked and whether that was approved. Deliveries made while masking was off appear with the MaskingOff policy.	Auditor, DbAdmin, ChangeManager, SystemManager
Sensitive Column Catalog	The state of the classification catalog and its learning loop: active rules with their scope and masking style, plus candidate columns awaiting a decision. Contains no data values.	-
Masking Exempt Servers	Servers whose masking regime has been relaxed: which mode, the written reason, who set it and when, plus how many deliveries were made from that server.	Auditor, DbAdmin, ChangeManager, SystemManager
Manual Data Modifications	Requests that modify data directly through DML or procedure calls.	-
Data Access and Export Log	Who downloaded a result file, for what stated reason and from which address, who exported an evidence dossier, when a result file was deleted after its retention period, and who opened a restricted report. This is the personal data access trail.	Auditor, DbAdmin, ChangeManager, SystemManager

20.7 Configuration & Compliance group

Report	What it shows	Restriction
Configuration Changes	Every change to the controls themselves: servers, parameters, approval policies, roles, SQL standards, critical objects, sensitive columns, patterns, mail sources and license events, with old and new values.	Auditor, DbAdmin, ChangeManager, SystemManager
Settings Approval Trail	Dual control evidence for settings: who requested a change, who approved or rejected it, how long the decision took and whether the same person did both.	Auditor, DbAdmin, ChangeManager, SystemManager

20.8 Reading suggestions

Goal	Reports to look at
Audit preparation	Control Exceptions, Approval Step Detail, Settings Approval Trail, User Permission Matrix
Personal data compliance	Query Result Activity, Masking Activity, Data Access and Export Log, Masking Exempt Servers
Process efficiency	Approval and Execution Duration, Work in Progress, Execution Activity
Technical debt	Object Change Frequency, Multi Release Objects, Active Conflicts
Access review	User Permission Matrix, User Access Changes, Privileged Account Operations

20.9 Exporting

Report results are shown as a table on screen and can be exported as a file. Opening a restricted report is written to the audit trail and appears in the Data Access and Export Log report.

Section 21. Notifications and Email

21.1 Which event notifies whom

Event	Recipient
Request created	Users holding the role of the relevant approval step.
Approval step became due	Users who can close that step. If the step is assigned to a person, that person.
Request approved	The requester.
Request scheduled	The requester.
Request executed	The requester. For query result requests, also the recipient list.
Request rejected	The requester, with the reason.
Approval step rejected	The requester, with the reason.
Request marked no longer needed	The requester.
Automatic execution failed	An error notification into the queue.
Release package failed	An error notification into the queue.
Worker unexpected error	A critical notification to the Workers:WorkerErrorMailTo address.

If a step is assigned to a user (the requester's direct manager meeting the step role), the notification goes to them first.

21.2 Email is never sent inside a transaction

Email is never sent inside a database transaction. The notification is written to a queue instead, and the worker drains the queue after the transaction completes.

The reason: if the SMTP server were slow or unreachable, the database transaction would stay locked. Also, if the transaction were rolled back the email would already have been sent, producing "a notification for an event that never happened".

What the user sees: after the screen confirms the action, the email arrives within a few seconds rather than instantly. The delivery interval is the Workers:NotificationFlushSeconds setting.

21.3 The Email Sources screen

Screen: Settings > Email Sources

Permission: settings.email

The pool of addresses usable in query result deliveries.

Field	What it shows
Email	The address.
Active	Is it in use.
Approved	Has the address been approved. Delivery to an unapproved address requires an extra approval step.
Approved by / Approved at	
Source	Whether it was added manually or by the system.

21.4 The Notification Logs screen

Screen: Settings > Notification Logs

Permission: settings.notificationlogs

Column	What it shows
Channel	Delivery channel (email).
Recipient	Target address.
Subject	Email subject.
Status	Sent, failed, queued.
Error	The reason if it failed.
Attempt count	How many times it was tried.
Time	

This is the first place to look when a notification did not arrive. Records are deleted after 90 days by a maintenance task.

Section 22. ITSM Ticket Number

22.1 What it is for

It links the ticket number from the organisation's existing change management system to the request, so that the SQL Change Guard record matches the corporate process record.

Screen: Settings > ITSM Tickets

Permission: settings.itsm

22.2 Fields

Field	What it does
Ticket No	The ticket number. It is unique.
Title	The subject of the ticket.
Link	An address to open the ticket directly.
Source	The system the ticket came from.

22.3 Requirement and format validation

Two parameters govern this:

Parameter	Effect
ChangeTicketRequired	When true, a request cannot be saved without a ticket number.
ChangeTicketRegex	Validates the format of the entered number. No check when empty.

Example: with ChangeTicketRequired true and ChangeTicketRegex set to ^CR-\d{5}$:

Entered	Result
CR-12345	Accepted.
CR-1234	Rejected, five digits are required.
12345	Rejected, the prefix is missing.
(empty)	Rejected, it is mandatory.

The ticket number appears in the request list, in reports and in the evidence dossier. Conflict reports also use the ticket number to find requests trying to change the same object under different tickets.

Section 23. Date and Time Model

23.1 Storage: always UTC

Every date field in the system is of type DATETIMEOFFSET and is stored in UTC. There are no exceptions.

Affected tables include requests, audit records, approval steps, servers, users, parameters, maintenance records, notification records, masking records and others. Fields such as CreatedAt, ModifiedAt, OccurredAt, ExecutedAt and ScheduledAt are all UTC.

23.2 Why UTC is used

There are three reasons:

1. Daylight saving time. A record stored in local time causes the same hour to occur twice when clocks go back. The question "what happened at 02:30" then has two different answers. UTC has no such ambiguity.

2. Users in different cities. A user in Istanbul and a user in London see the same event at different clock times, but the record is single and consistent.

3. Ordering. The audit chain needs one single time axis to be ordered correctly.

23.3 Display

Place	**Which time is shown**
Web interface	The browser's local time. The user configures nothing, it converts automatically.
Server-side documents (evidence dossier PDF, email text)	The time zone in the Display:TimeZone setting. Default Europe/Istanbul.
Reports	Dates in report output are converted from UTC to the display time zone.
Audit log	Local time on screen, UTC in the database.

The browser time cannot be known for server-side documents, which is why a central time zone setting is used.

If an invalid time zone is entered: the application does not stop producing documents. It falls back to UTC and writes a warning to the log. Documents also display the time zone label, so the reader can see which zone was applied. Producing a document and checking that label after installation is recommended.

23.4 A concrete example

A request was executed on 2 August 2026 at 12:00 UTC.

Who is looking	What they see
A user in Istanbul (UTC+3)	2 August 2026, 15:00
A user in London (summer time, UTC+1)	2 August 2026, 13:00
The evidence dossier PDF (Display:TimeZone = Europe/Istanbul)	2 August 2026, 15:00
The database record	2026-08-02 12:00:00 +00:00

All three views show the same instant. The record is single.

23.5 Local day boundaries in filters

When a user filters for "2 August", they mean the start and end of their own local day.

For a user in Istanbul, 2 August is:

- Start: 1 August 21:00 UTC

- End: 2 August 21:00 UTC

The system performs this conversion automatically. The user never has to do UTC arithmetic.

This detail matters in reports: two people running the same report from different time zones may see slightly different record sets even when selecting the same date. This is not an error; both are correctly seeing their own day.

Section 24. Language and Localisation

24.1 Interface language

The web interface works in Turkish and English. The language is selected in the interface and the user can change it at any time. All labels, messages and help texts are held in resource files.

24.2 The language of API messages

API messages are always returned in English. The canonical language of the API is fixed; the client's language preference does not change the API response.

The reason: API responses go into logs, audit records and integrations. If their language changed with the caller, the same event would be recorded in two different languages, making search and comparison impossible.

The interface maps the response to its own language resources and shows it in the language the user selected. So the user sees a Turkish message while the text recorded in the background is English.

24.3 Report output

Report names, descriptions and column headings are in English. The reason is given in Section 20.1.

24.4 Widely accepted terms

Widely accepted technical terms such as role names are not translated. DbAdmin, ChangeManager and SystemManager are written the same way in both languages. Translating them would make communication between teams harder.

Section 25. Maintenance and Operations

25.1 The Maintenance Tasks screen

Screen: Settings > Maintenance

Permission: settings.maintenance

Field	What it does	Default
Name	Name of the task.	
Description	What it does.	
Procedure Name	The stored procedure to run. A name beginning with INTERNAL: is an in-application task.	
Frequency (minutes)	How often it runs.	
Timeout (seconds)	The longest it may take.	
Retry Count	How many times it is retried on error.	
Retry Delay (seconds)	The wait between retries.	
Enabled	Should the task run.	On
Internal	A task shipped with the product that cannot be deleted.	

25.2 Default maintenance tasks

Task	What it does	Frequency	Timeout
Expired Token Cleanup	Deletes expired session refresh records.	60 minutes	60 seconds
Old Notification Log Cleanup	Deletes notification records older than 90 days.	1440 minutes (daily)	120 seconds
QR File Cleanup	Deletes query result files past their retention period from disk and clears the path in the database. Driven by the QrFileRetentionDays parameter.	In-application	-

25.3 Maintenance logs

Every run is recorded: start, finish, duration, status, error message and attempt number. A task that keeps failing is visible in this list.

25.4 The Diagnostics screen

Screen: Settings > Diagnostics

Permission: settings.diagnostics

Check	What it verifies	If there is a problem
Mail configuration	The SMTP settings the application actually uses. The password is not shown.	Confirm the settings match what you expect.
Mail test	Sends a test email with the given or the current settings.	The error message comes from the SMTP server; check the server name, port, credentials and SSL setting.

25.5 Worker monitoring

The worker service reports its own status to the API at regular intervals. The interface shows whether the worker is online and when it last reported.

If the worker shows offline:

1. Check that the Windows service is running.

2. Check that WorkerApi:BaseUrl is correct.

3. Check that WorkerApi:Key matches Workers:ApiKey on the API side.

4. Look at the worker log files.

25.6 Backup recommendations

What to back up	Why	Frequency
Application database	All requests, rules and audit records.	Per your corporate policy.
Immutable database	The second copy of the audit trail. It must be backed up together with the main database; otherwise the comparison cannot be made.	Same as the main database.
Encryption keys	Security:ServerPasswordKey, Security:QrZipPasswordKey, Audit:HmacKeys, Jwt:SigningKey.	Whenever they change.

What to back up	Why	Frequency
License file	Needed on a reinstall.	Whenever it changes.
appsettings files	The whole configuration.	Whenever they change.

The most critical warning: encryption keys are not inside the database backup. If you back up only the database and lose the keys, you cannot recover server passwords and cannot verify old audit records. Keep the keys separately and securely from the database backup.

25.7 The crash recovery threshold

When a worker starts executing a request it claims that request and records the moment it started. If the service stops mid-run (a crash, a restart, a server shutdown) the request stays in Executing and nobody finishes it.

At the start of every cycle the worker looks for these stuck requests and returns them to Approved. The test is a single question: how long has this request been claimed?

Setting	Meaning
Workers:StaleExecutionRecoveryMinutes	An execution running longer than this is presumed dead.
Default	120 minutes
Lower bound	5 minutes. A smaller value is ignored and 120 minutes is used.
Upper bound	365 days. A larger value is capped at 365 days.

When a value outside these bounds is configured, a warning is written to the log at worker startup.

Why the threshold exists

Without a threshold, every claimed request would be presumed dead at the start of each cycle. With a single worker instance this causes no harm, because cycles never overlap. But if a second worker instance were started, its cycle could treat a request the first instance is currently executing as dead and return it to Approved. The request could then be picked up again and the same script could run twice.

How to choose the value

The threshold must be greater than the longest legitimate execution in your organisation.

Situation	Consequence
Threshold too small	A request that is still running may be presumed dead.

Situation	Consequence
Threshold too large	Recovery of a genuinely crashed request is delayed and the request shows as Executing for that period.

Example: if your longest script takes about 20 minutes, 45 minutes is a reasonable value. Recovery happens within 45 minutes and no running work is affected.

If you do not want to wait for recovery you can end the request by hand: cancel the execution on the stuck request or mark the request as manual.

Section 26. Troubleshooting

26.1 Warning and error message dictionary

The tables below list the messages a user may encounter, where they appear, why, and how to resolve them.

Permission messages

Message (EN)	Message (TR)	Where	Cause	Fix
You don't have permission for this action.	Bu işlem için yetkiniz yok.	Everywhere	Your role lacks the required permission.	Ask your administrator for the permission.
You don't have permission to create requests.	Yeni talep oluşturma yetkiniz yok.	Request screen	The cr.add permission is missing.	Have your role checked.
You don't have permission to edit this request.	Bu isteği düzenleme yetkiniz yok.	Request detail	You are not the owner.	Contact the request owner.
You don't have execute permission.	Script çalıştırma yetkiniz yok.	Request detail	The cr.execute permission is missing.	Ask an authorised user to execute it.
Approval permission is required.	Bu işlem için onay yetkisi gereklidir.	Request detail	The cr.approve permission is missing.	Route it to an authorised approver.
You don't have permission to download this QR file.	QR dosyasını indirme yetkiniz yok.	Request detail	QrRequesterCanDownload is off or your role is below QrDownloadMinimumRole.	See Section 7.7.
The auditor role cannot be granted approve or execute permissions.	Denetçi rolüne onaylama veya çalıştırma yetkisi verilemez.	Roles screen	The auditor cannot enter the flow for segregation of duties reasons.	Use a different role.
This account has been closed.	Bu hesap kapatılmıştır.	Sign-in screen	The user has been deleted.	Contact your administrator.

State messages

Message (EN)	Message (TR)	Cause	Fix
Only Pending requests can be approved.	Sadece Pending durumundaki istekler onaylanabilir.	The request is already approved or closed.	Refresh the page.

Message (EN)	Message (TR)	Cause	Fix
Only Approved requests can be executed.	Sadece Approved durumundaki istekler çalıştırılabilir.	The request is not approved yet.	Wait for the approval process to finish.
Only Approved requests can be scheduled.	Sadece Approved durumundaki istekler zamanlanabilir.	Same cause.	
Completed or cancelled requests cannot be edited.	Tamamlanmış veya iptal edilmiş istekler düzenlenemez.	The request is terminal.	Open a new request or copy the existing one.
Cannot perform '{1}' in status: {0}.	Bu durumda ({0}) {1} yapılamaz.	The action does not fit the current state.	See the transition matrix in Section 11.1.
Only scheduled Approved requests can have their schedule cancelled.	Sadece zamanlanmış Approved isteklerin zamanlaması iptal edilebilir.	The request has no schedule.	

Segregation of duties messages

Message (EN)	Message (TR)	Cause	Fix
The requester cannot approve their own request.	İstek sahibi kendi isteğini onaylayamaz.	RequesterCanSelfApproveExecute is off.	Route it to another approver.
The requester cannot execute their own script.	İstek sahibi kendi scriptini çalıştıramaz.	The same setting.	Another authorised user must execute it.
The person who approved this request cannot execute it.	Bu talebi onaylayan kişi aynı talebi çalıştıramaz.	ApproverCanExecute is off.	A different authorised user must execute it.

Rollback messages

Message (EN)	Message (TR)	Cause	Fix
A rollback record is required before execution on this server.	Bu sunucu için çalıştırmadan önce rollback kaydı zorunludur.	"Rollback Required" is on for the server and no rollback exists.	Create a rollback request.
An active rollback already exists for this request.	Bu talep için zaten aktif bir rollback kaydı var.	A second rollback attempt.	Use the existing rollback.

Message (EN)	Message (TR)	Cause	Fix
Rollback can only be created by the owner of the request.	Rollback yalnızca talebin sahibi tarafından oluşturulabilir.	You are not the owner.	Ask the request owner.
The source request has not been executed yet.	Kaynak talep henüz çalıştırılmadı.	A rollback cannot run before the source is applied.	Execute the source request first.
Rollback can only be generated for Change type requests.	Rollback sadece Change tipindeki talepler için oluşturulabilir.	Query result requests have no rollback.	-
Rollback type requests cannot be copied.	Rollback tipi talepler kopyalanamaz.		Copy the source request instead.
The rollback request server must match the source request server.	Rollback talebi kaynak talebin sunucusuyla eşleşmelidir.	The server was changed.	Select the same server.
The existing rollback was superseded because the source request scripts were edited.	Kaynak talebin scriptleri düzenlendiği için mevcut rollback geçersiz kılındı.	The source changed and the old rollback no longer fits.	A current rollback is created on re-approval; create it manually if auto-generation is off.

Release package and sandbox messages

Message (EN)	Message (TR)	Cause	Fix
This request is under release package '{0}'. Remove it from the package first.	Bu talep '{0}' yayın paketi altında. Önce paketten çıkarın.	The request is in an active package.	Remove it from the package or proceed through the package.
Only change requests can be added to a release package.	Sürüm paketine yalnızca değişiklik talepleri eklenebilir.	A query result request was added.	See Section 15.6.
Query result requests cannot be executed inside a release package.	Sorgu sonucu talepleri release paketi içinde çalıştırılamaz.	Masking and records are produced only in the dedicated flow.	Execute the request on its own.
Oracle servers are not currently supported in release packages.	Oracle sunucuları şu an release paketlerinde desteklenmemektedir.	A known limitation.	Execute Oracle requests individually.
Sandbox is only available for 'Change' type requests.	Sandbox yalnızca 'Change' tipindeki talepler için geçerlidir.		-

Settings approval messages

Message (EN)	Message (TR)	Cause	Fix
The change was sent for approval.	Değişiklik onaya gönderildi.	The four eyes rule is in force.	Ask an authorised colleague to approve.
There is already a pending change for this record.	Bu kayıt için onay bekleyen bir değişiklik zaten var.	A second request.	Resolve the existing one first.
A decision has already been made for this change.	Bu değişiklik için karar zaten verilmiş.	Someone decided before you.	Refresh the list.
You cannot approve or reject a settings change you requested yourself.	Kendi talep ettiğiniz ayar değişikliğini onaylayamaz veya reddedemezsiniz.	The four eyes rule.	Another user with the same permission must decide.
A rejection reason is required.	Reddetme gerekçesi zorunludur.	The reason is empty.	Write a reason.
The change was approved but could not be applied: {0}	Değişiklik onaylandı ancak uygulanamadı: {0}	The command failed at approval time.	Read the reason, fix the issue and request again.
The four eyes rule needs at least two users holding the parameters permission.	Dört göz kuralı için parametre yetkisine sahip en az iki kullanıcı gerekir.	Only one authorised user exists.	Define a second authorised user.

Server and license messages

Message (EN)	Message (TR)	Cause	Fix
Server is not active: {0}	Sunucu aktif değil: {0}	The server has been deactivated.	Activate it or pick another server.
This server's database type ({0}) is not covered by your license.	Bu sunucunun veritabanı tipi ({0}) lisansınız tarafından desteklenmiyor.	Outside the license scope.	Update your license.
The selected server type ({0}) does not match the request server type ({1}).	Seçilen sunucunun veritabanı tipi ({0}) talep sunucusu tipi ({1}) ile uyuşmuyor.	Servers of different types were mixed.	Select servers of the same type.

Request content messages

Message (EN)	Message (TR)	Cause	Fix
A request must have at least 1 script.	Talep en az 1 script içermelidir.	An empty request.	Add a script.
QueryResult requests can only have 1 script.	QueryResult tipi talepler için yalnızca 1 script eklenebilir.	Multiple scripts were attempted.	Combine the query into a single script.
Earliest schedule date: {0} (minimum delay: {1} seconds).	En erken zamanlama tarihi: {0} (minimum {1} saniyelik gecikme).	The selected time is too close.	Pick a later time. See Section 6.4.
A reason is required when a mode other than full masking is selected.	Tam maskeleme dışında bir rejim seçtiğinizde gerekçe zorunludur.	The exemption reason is empty.	Enter a written reason.

Execution log notes

Note	Meaning
-- {0} block(s) executed successfully; execution STOPPED with an ERROR at block {1}.	Shows how far it ran and where it stopped.
-- ORACLE ATOMICITY NOTE: ...	On an Oracle target, DDL statements executed before the error may be permanent. Review the residual state and run the rollback request manually if needed.
-- Detection time budget exceeded; unscanned columns were masked.	Sensitive data detection exceeded its time budget; unscanned columns were masked on the safe side.

26.2 Common scenarios

Cannot sign in

Check	What to do
Is the account locked	Consecutive failed attempts lock the account. Wait Auth:AccountLockoutMinutes or have an administrator unlock it.
Has the account been deleted	If you see "this account has been closed", an administrator must revive it.
LDAP or local	For an LDAP account the password changes in the corporate directory.

Check	What to do
Two-step verification	If two-step verification is set up on your account you must enter a code from the authenticator app. If it is not set up, no code is asked. See Section 2.4.9.

Cannot connect to a server

Check	What to do
Connection test	Test from Settings > Servers and read the message.
Network and port	Check the firewall and the port number.
Credentials	The password may have changed.
Encryption key	If Security:ServerPasswordKey changed, old passwords cannot be decrypted. Re-enter the password.
License	Is the database type within the license.

A request does not run

Check	What to do
Status	Is the request Approved.
Pending approval step	Look at the request detail to see which step it waits on.
Schedule	If scheduled, the time may not have come yet.
Rollback requirement	If "Rollback Required" is on for the server, create a rollback.
Never-skip finding	With such a finding automatic execution does not happen; run it manually.
Worker	Is the worker running.
Active package	The request may be inside a release package.

The worker does not run

Check	What to do
Service status	Check the Windows service.

Check	What to do
MainWorkEnabled	The flag may be false.
API key	Do WorkerApi:Key and Workers:ApiKey match.
Logs	Look at the worker log files.

A report comes back empty

Check	What to do
Date range	The default is the last 30 days. Widen the range.
Server filter	The wrong server may be selected.
Role restriction	You may lack permission for a restricted report.
Module	Is the InsightsReporting module licensed.
Data	There may genuinely be no records in that period, which can also be a good result.

Email is not delivered

Check	What to do
Mail:Enabled	It may be off.
Mail:DevMode	If on, all email goes to the test address.
Notification logs	Read the error message on Settings > Notification Logs.
Mail test	Send a test email from Settings > Diagnostics.
Worker	Notifications are sent by the worker; is it running.

Section 27. Appendices

27.1 Enum glossary

Roles (enmRole)

Value	Name	Hierarchy level
0	NoAccess	0
5	Viewer	5
10	Auditor	10
15	Requester	15
20	RequestManager	20
25	DbAdmin	25
30	ChangeManager	30
35	SystemManager	35

Request status (ChangeRequestStatus)

Value	Name	Terminal
1	Pending	No
2	Approved	No
3	Executed	Yes
4	Rejected	Yes
5	Manual	Yes
6	NoLongerNeeded	Yes
7	Executing	No

Request type (QueryType)

Value	Name
0	None
1	Change
2	QueryResult

Value	Name
3	Rollback

Database type (DatabaseType)

Value	Name
1	SqlServer
2	Oracle
3	PostgreSQL

Risk band (RiskTier)

Value	Name	Score range
0	Info	90 - 100
1	Low	70 - 89
2	Medium	50 - 69
3	High	25 - 49
4	Critical	0 - 24

Release package execution mode (ReleaseExecutionMode)

Value	Name
1	AllOrNothing
2	StopOnError
3	ContinueOnError

Approval step status

Value	Meaning
0	Pending

Value	Meaning
1	Approved
2	Rejected
3	Skipped

Server environment

Value
Production
Staging
Development
Test

Priority in approval policy resolution: Production > Staging > Test > Development > undefined.

Masking regime

Value	Meaning
Strict	Catalog and patterns together.
ContentOnly	Patterns only.

Masking policy on the record

Value	Meaning
AutoMask	Detected columns were masked.
WarnOnly	Detection ran, nothing was masked.
MaskingOff	The master switch was off, no detection ran.

Masking reason types

Value	Meaning
Catalog	The column name matched the catalog.
Pattern	The value matched a pattern.
UnmaskApproved	An approved exception.
NotScanned	Not scanned because of the time budget, masked on the safe side.
PolicyWarnOnly	The policy is WarnOnly.
MaskingOff	The master switch is off.

Candidate column status

Value	Meaning
New	Awaiting a decision.
Accepted	Added to the catalog.
Ignored	Dismissed.

Settings change status

Value	Meaning
Pending	Awaiting a decision.
Approved	Approved and applied.
Rejected	Rejected.
Failed	Approved but could not be applied.

License status

Value	Meaning	Restricted mode
Valid	Valid.	No
Trial	Trial.	No

Value	Meaning	Restricted mode
Expiring	Close to expiry.	No
GracePeriod	Inside the grace period.	No
Expired	Expired.	Yes
Invalid	Invalid.	Yes
Missing / NoLicense	Absent.	Yes

Audit hash version

Value	Algorithm	Scope
1	SHA-256	Old value and IP excluded.
2	SHA-256	Old value and IP included.
3	HMAC-SHA256	Version 2 scope, with a secret key.

27.2 Parameter quick reference

Key	Group	Default	Control setting
DefaultQueryTimeout	General	30	No
MaxQueryTimeout	General	3600	No
ExecutionDelaySeconds	General	300	No
RejectOnError	General	true	Yes
RequesterCanSelfApproveExecute	Segregation of Duties	true	Yes
ApproverCanExecute	Segregation of Duties	true	Yes
RequireDistinctApproverPerStep	Segregation of Duties	false	Yes
EmergencyMinimumRole	Segregation of Duties	Requester	Yes
SettingsApprovalPolicy	Settings Approval	Disabled	Yes
ChangeTicketRequired	ITSM Ticket	true	Yes
ChangeTicketRegex	ITSM Ticket	^CR-\d{5}$	No
QrFileRetentionDays	Query Result	1	Yes
QrRequesterCanDownload	Query Result	true	Yes

Key	Group	Default	Control setting
QrDownloadMinimumRole	Query Result	ChangeManager	Yes
QrMaskingEnabled	Sensitive Data	true	Yes
QrSensitiveDataPolicy	Sensitive Data	AutoMask	Yes
QrSensitiveDetectionSampleSize	Sensitive Data	1000	Yes
QrSensitiveDetectionThresholdPct	Sensitive Data	20	Yes
QrSensitiveDetectionSampleMode	Sensitive Data	Random	Yes
QrSensitiveDetectionTimeBudgetSeconds	Sensitive Data	10	Yes
QrColumnPreviewEnabled	Sensitive Data	true	No
UnmaskedColumnRequestPolicy	Sensitive Data	RequireApproval	Yes
UnmaskedColumnApproverRole	Sensitive Data	SystemManager	Yes
QrEmailApprovalPolicy	Email Approval	RequireApproval	Yes
QrEmailApproverRole	Email Approval	ChangeManager	Yes
QrEmailAutoApproveUsers	Email Approval	True	Yes

27.3 Permission key quick reference

Key	Screen or action it enables
cr.add	Creating requests
qr.add	Creating query result requests
cr.execute	Execution
cr.sandbox	Sandbox
cr.approve	Approving, rejecting, scheduling, marking manual
cr.seeall	Seeing all requests
report.view	Reports
dashboard.view	Dashboard
audit.view	Audit log and verification actions
audit.evidenceexport	Exporting evidence dossiers
rp.manage	Release packages
settings.servers	Servers

Key	Screen or action it enables
settings.users	Users
settings.roles	Roles
settings.params	Parameters
settings.approvals	Approval policies
settings.standards	SQL standards
settings.critical	Critical objects
settings.itsm	ITSM tickets
settings.qrsensitive	Sensitive columns and candidate columns
settings.patterns	Sensitive data patterns
settings.email	Email sources
settings.maintenance	Maintenance tasks
settings.notificationlogs	Notification logs
settings.license	License
settings.diagnostics	Diagnostics

27.4 UTC storage note

All date fields of the following tables are of type DATETIMEOFFSET and stored in UTC:

Users, ChangeRequests, ChangeRequestApprovalSteps, ChangeRequestResultColumns, AuditLogs, RefreshTokens, Servers, SqlStandards, CriticalObjects, EmailSources, ReleasePackages, Parameters, Scripts, ScriptObjects, ItsmTickets, QrSensitiveColumns, ReportDefinitions, ReportParameters, MaintenanceTasks, MaintenanceTaskLogs, NotificationLogs, ApprovalPolicies, ApprovalPolicySteps, ServerApprovalPolicies, SensitiveDataPatterns, QrMaskingAuditLog, QrMaskingAuditDetail, SensitiveColumnCandidates, SettingChangeRequests, SandboxResults, RiskCalc_Result.

Display rules are in Section 23.

27.5 Installation checklist

[ ] An empty database has been created.

[ ] ConnectionStrings:Default has been filled in.

[ ] Jwt:SigningKey has been generated and written.

[ ] Security:ServerPasswordKey has been generated and written.

[ ] Security:QrZipPasswordKey has been generated and written.

[ ] Audit:HmacKeys and Audit:ActiveKeyId have been defined.

[ ] Mail settings are configured and DevMode is false.

[ ] Display:TimeZone matches the organisation.

[ ] The web interface address has been added to AllowedOrigins.

[ ] The QueryResult:OutputPath folder exists and is writable.

[ ] Workers:ApiKey matches WorkerApi:Key on the worker side.

[ ] The API has been started and the schema and default data were created.

[ ] The license has been uploaded.

[ ] The admin password has been changed.

[ ] The worker service has been installed and started.

27.6 Initial configuration checklist

[ ] Servers are defined and their connection tests succeed.

[ ] The critical object list has been entered.

[ ] SQL standards have been reviewed and unnecessary ones disabled.

[ ] The sensitive column catalog and patterns have been reviewed.

[ ] At least one global approval policy is defined.

[ ] Environment-specific policies are defined.

[ ] An emergency policy is defined if it will be used.

[ ] Users have been added and their roles assigned.

[ ] At least two users hold each settings permission.

[ ] SettingsApprovalPolicy has been set to a value suitable for the organisation.

[ ] ChangeTicketRegex matches the organisation's ticket format.

[ ] The segregation of duties parameters have been decided.

[ ] A test request has been opened and taken end to end.

27.7 Audit preparation checklist

[ ] The Control Exceptions report has been run and every row can be explained.

[ ] Dual control evidence has been taken from the Approval Step Detail report.

[ ] Control changes have been reviewed in the Settings Approval Trail report.

[ ] The User Permission Matrix report has been taken.

[ ] The Data Access and Export Log report has been reviewed.

[ ] The Masking Activity report has been checked for MaskingOff records.

[ ] Reasons in the Masking Exempt Servers report are current.

[ ] Verify Chain has been run and is clean.

[ ] Verify Immutable has been run with no mismatched records.

[ ] Evidence dossiers have been produced and verified for sample requests.