

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	19
0.1 What SQL Change Guard is	19
0.2 Three focus areas	20
0.3 Who uses it	20
0.4 Glossary	20
Section 1. Architecture and Components	23
1.1 Overview	23
1.2 Components and their duties	24
1.3 Supported database types	24
1.4 Worker jobs	25
1.5 Scaling and deployment	26
1.6 Measured capacity and sizing	27
How to read this table, and the recommended server	27
Storage planning	28
Concurrent use and team size	28
Number of servers	29
Summary of limits	30
Section 2. Installation	31
2.1 Prerequisites	31
2.2 Installation steps	31
2.3 Creating the database	32
2.4 appsettings.json field by field	33
2.4.1 ConnectionStrings	35
2.4.2 Database:EvidenceLedger and Evidence:Ledger	35
2.4.3 Auth	36
2.4.4 Jwt	38
2.4.5 Security (encryption keys)	39
2.4.6 License	39
2.4.7 Mail	40

2.4.8 Ldap.....	40
2.4.9 Other settings.....	43
2.4.10 SIEM (SOC integration).....	47
2.4.11 Sso (corporate sign-in)	51
2.5 Worker side appsettings.json	55
2.6 First administrator and first sign-in	55
2.7 Installation verification checklist.....	56
2.8 Common installation problems.....	56
Section 3. Licensing	57
3.1 What the license file is	57
3.2 License content	57
3.3 Modules	57
3.4 Server limit	58
3.5 Database types come from the license.....	59
3.6 License states and system behaviour.....	59
3.7 Uploading a license	59
3.8 License error messages	60
Section 4. Roles, Permissions and Hierarchy	61
4.1 Roles and their numeric values.....	61
4.2 Id and hierarchy are separate concepts.....	62
4.3 The special position of the Auditor role	62
4.4 Role permission matrix	62
4.5 Permission keys	64
4.6 Hiding in the interface is not security	65
4.7 User lifecycle	65
When a permission change takes effect	66
4.8 LDAP users versus local users.....	67
Users who sign in with a corporate account (SSO)	69
Role source: corporate account (SSO) and directory (LDAP) together	72

When a person cannot sign in with the corporate account or the directory	73
4.9 Password policy and BCrypt	75
What a hash is	75
What a salt is	76
Why BCrypt was chosen	76
4.10 The Users screen	76
4.11 The Roles screen	77
4.12 Bulk user import	79
Preparing the file	79
Validation and error reporting	79
An imported account cannot sign in	80
Audit trail	80
Section 5. Security and Encryption Model	80
5.1 Three different needs, three different methods	80
5.2 Server passwords: AES-256-GCM	81
What symmetric encryption is	81
Why GCM was chosen	81
Where it is used	81
5.3 What is protected where	81
5.4 Audit trail: append-only ledger	82
What the ledger is	82
The digest and the external witness	82
Why there is no secret key	82
5.5 Database level write log	82
Why the write log has no seal	83
What to do at installation	83
5.6 Secret masking	83
Section 6. Server Definitions	85
6.1 What this screen is for	85

6.2 Field by field	86
6.3 Connection test	91
6.4 Automatic execution: what happens behind the scenes	91
The chain	91
The conditions	92
How the time is computed	92
Who the record names	93
6.5 Masking regime	93
Several servers can be selected only on a query result (QueryResult) request; change and rollback requests go to a single server, and the backend enforces this as well. The reason is that there is no distributed transaction across servers: if the first server committed and the second failed, a partial change would remain. Server names are stored in a single field, separated by semicolons:	94
6.7 Deleting versus deactivating a server	94
6.8 Session name on the target and the session probe	95
Section 7. Parameters	96
7.1 What this screen is for	96
7.2 Fields on the screen	96
7.3 General group	96
DefaultQueryTimeout	96
MaxQueryTimeout	96
ExecutionDelaySeconds	97
RejectOnError	97
7.4 Segregation of Duties group	97
RequesterCanSelfApproveExecute	98
ApproverCanExecute	98
RequireDistinctApproverPerStep	98
EmergencyMinimumRole	98
7.5 Settings Approval group	99
SettingsApprovalPolicy	99

7.6 ITSM Ticket group	99
ChangeTicketRequired	99
ChangeTicketRegex	100
EmergencyTicketRequired	100
EmergencyTicketRegex	100
Tickets on rollback requests.....	101
RequestTicketRegex	101
ITSM integration parameters	101
7.7 Query Result group	103
QrFileRetentionDays	103
QrRequesterCanDownload	103
QrDownloadMinimumRole	104
QrDeliveryMode.....	104
QrMaxRowCount.....	105
QrOutputFormat	105
QrCsvThresholdRowCount	106
QrExcelAutoFitMaxRows.....	106
7.8 Sensitive Data group	107
QrMaskingEnabled.....	107
QrSensitiveDataPolicy	107
QrSensitiveDetectionSampleSize	107
QrSensitiveDetectionThresholdPct	108
QrSensitiveDetectionSampleMode.....	108
QrSensitiveDetectionTimeBudgetSeconds	108
QrColumnPreviewEnabled	108
UnmaskedColumnRequestPolicy	109
UnmaskedColumnApproverRole.....	109
7.9 Email Approval group.....	109
QrEmailApprovalPolicy.....	109

QrEmailApproverRole	109
QrEmailAutoApproveUsers	110
7.10 When a parameter change takes effect	110
7.11 SIEM group	111
Events that cannot be silenced by the threshold	112
Narrowing the scope is announced to the SOC	113
Section 8. Settings Change Approval (Four Eyes)	113
8.1 What it is for	113
8.2 Scope	113
What a control setting means	114
8.3 The flow	114
8.4 What replaying the command means	115
8.5 Who approves	116
8.6 The deadlock gate	116
8.7 The Settings Approvals screen	116
8.8 Messages on this screen	117
Section 9. Critical Objects, SQL Standards and Sensitive Data	118
9.1 Critical Objects	118
What it is for	118
Fields	118
Matching logic	118
What changes when a critical object is detected	118
9.2 SQL Standards	119
What it is for	119
Fields	119
What happens on a violation	120
Example standards	120
9.3 Sensitive Data Patterns	121
What it is for	121

Fields	121
Why validators exist	122
Partial masking example	122
9.4 Sensitive Column Catalog.....	122
What it is for.....	122
Fields	122
9.5 Candidate Columns (the learning loop)	123
How a candidate appears.....	123
Columns	123
What happens on acceptance	124
Section 10. Approval Policies	125
10.1 What a policy is	125
10.2 Policy fields	125
10.3 Policy mapping (scope)	125
What Priority does	126
10.4 Policy selection flow.....	126
What happens when several policies match	127
What happens when none matches	127
What happens when every step is skipped	128
10.5 Step fields and behaviour	128
How a step decides	128
Example policy	129
10.6 Segregation of duties rules	130
The role-skip trace	130
10.7 Deadlock scenarios	130
10.8 Emergency availability	130
Section 11. Change Request End to End	132
11.1 The state machine.....	132
Transition matrix	132

11.2 The request creation screen	134
Prerequisite: the server must have an approval policy	136
11.3 The script editor and parsing	136
When parsing runs	136
What parsing produces	137
Parsing errors	137
Affected objects and the difference view	137
11.4 The request detail screen.....	138
Header fields	138
The scripts and objects table	139
The approval steps table	140
Execution result.....	141
Audit / event list.....	141
11.5 Buttons and enablement conditions.....	141
11.6 The execution flow.....	142
Scheduled execution	143
Cancelling an execution	143
Crash recovery.....	144
Approval chain: what was saved, approved and executed is the same.....	144
11.7 Partial success and rollback	145
11.8 Rollback requests	145
How the rollback script is produced	146
What the rebuild covers.....	146
The two different warnings on screen	147
Consequences of an incomplete rollback script	147
11.9 Execution attribution	148
11.10 Emergency override	148
11.11 The request list screen	149
Section 12. Risk Band	150

12.1 Purpose and principles.....	150
12.2 What a band is	150
12.3 How the band is determined	150
12.4 How the score is computed (for the record).....	151
12.5 Worked example: band High with three High findings	151
Variations of the same example.....	152
12.6 The score explanation (ScoreExplain)	153
12.7 Never-skip rules	153
12.8 Where the risk band appears.....	154
12.9 How the band affects decisions	154
Section 13. Query Result Flow	155
13.1 What it is for.....	155
13.2 Query types.....	155
13.3 Creating a query result request	155
Column preview	156
The email approval step.....	157
The unmasked column approval step	157
13.4 What happens behind the scenes at execution.....	157
How the masking decision is made.....	158
What is recorded.....	159
13.5 The result file and delivery.....	160
Download permission	160
13.6 Special rules in this flow.....	161
13.7 Result file format and row limit	162
13.8 Measured cost and performance.....	162
Section 14. Sandbox.....	164
14.1 What it is for.....	164
14.2 How it works	164
14.3 Effect on the flow	165

14.4 Known limits.....	165
14.5 Sandbox request and window events	166
Section 15. Release Package	166
15.1 What it is for.....	166
15.2 Package fields.....	166
15.3 Execution modes	166
15.4 Package content and ordering	167
15.5 Differences from the single request flow.....	167
15.6 Constraints	168
15.7 Window in package execution	169
Section 16. Object History (DDL History)	169
16.1 What it is for.....	169
16.2 Screen flow.....	169
16.3 Audit value	170
Section 17. Audit Trail and Integrity.....	172
17.1 What an audit record holds	172
17.2 What is not held.....	172
17.3 Event naming convention	172
17.4 The old and new value contract.....	176
17.5 The rule set snapshot ("what applied that day").....	176
17.6 Ledger Status.....	176
Verify Script Integrity	177
17.7 Scope summary of the write log	178
17.8 Delivering the evidence outside the organisation	178
17.9 Timestamping (optional).....	178
17.10 The Audit Log screen.....	180
17.11 The audit records package: delivering the raw records.....	181
17.12 SOC integration (SIEM).....	182
When events are sent	182

Delivery resumes where it stopped	182
Heartbeat	183
What is sent and what is not	183
17.13 How to read the message that reaches the SOC	183
A sample message.....	183
The priority number and local0	184
How the severity is decided	186
The message that reaches the SOC when the scope is narrowed	187
17.14 Execution and sandbox window events.....	188
17.15 Carrying the window events to the SIEM	191
17.16 Evidence ledger and evidence write boundary	192
17.17 Ledger digest.....	193
17.18 Period package.....	194
Section 18. Evidence Dossier	194
18.1 What it is for.....	194
18.2 Who produces it.....	194
18.3 File content	195
18.4 Why query result data is not included	196
18.5 The export itself is audited	197
18.6 How the file is verified	197
What a failed verification means	198
Verifying without touching the system	198
The verification script inside the package.....	199
18.7 Very long scripts and the PDF	200
18.8 Sessions of the decision makers	200
18.9 What an auditor can prove with this file	201
18.10 Execution windows and exceptions	203
Section 19. Dashboard	203
19.1 The time separation	203

19.2 Range cards	204
19.3 Now cards.....	204
19.4 Execution success rate	204
19.5 Charts and lists	205
19.6 The exceptions card	205
19.7 Differences by role	206
19.8 Empty data	206
19.9 The explanation marks on the dashboard	207
19.10 The worker indicator and outage records.....	207
Outages are written to the audit log.....	208
Section 20. Reports	208
20.1 General rules	208
20.2 Common parameters	209
20.3 Change Governance group.....	209
20.4 Object & Structure group.....	212
20.5 Access & Authorization group.....	213
The Expected Access Principals report	213
20.6 Data & Privacy group	214
20.7 Configuration & Compliance group	215
20.8 Reading suggestions.....	216
20.9 Exporting	216
Section 21. Notifications and Email	217
21.1 Which event notifies whom	217
21.2 Email is never sent inside a transaction.....	217
21.3 The Email Sources screen.....	217
21.4 The Notification Logs screen	218
Section 22. ITSM Integration and Ticket Number	219
22.1 What it is for.....	219
22.2 Fields	219

22.3 Requirement and format validation.....	219
22.4 When a row is added to the ticket list	221
22.5 ITSM integration.....	221
22.6 Record kinds.....	222
22.7 How the state in ITSM is read	222
22.8 When a request is saved	223
22.9 Right before execution	224
22.10 When ITSM cannot be reached	225
22.11 Notes written to ITSM	225
22.12 The ITSM screen.....	226
22.13 Setup	226
22.14 Points to watch.....	227
Section 23. Date and Time Model.....	227
23.1 Storage: always UTC.....	227
23.2 Why UTC is used.....	227
23.3 Display	228
23.4 A concrete example.....	228
23.5 Local day boundaries in filters	228
Section 24. Language and Localisation	230
24.1 Interface language	230
24.2 The language of API messages.....	230
24.3 Report output	230
24.4 Widely accepted terms	230
Section 25. Maintenance and Operations	231
25.1 The Maintenance Tasks screen	231
25.2 Default maintenance tasks.....	231
25.3 Maintenance logs.....	232
25.4 The Diagnostics screen.....	232
25.5 Worker monitoring.....	232

25.6 Backup recommendations	233
25.7 The crash recovery threshold.....	233
Why the threshold exists	234
How to choose the value	234
Execution window during recovery.....	235
25.8 Watching SIEM delivery	235
Common situations	235
25.9 Verifying the SOC connection without a collector	236
Section 26. Troubleshooting	238
26.1 Warning and error message dictionary	238
Permission messages	238
State messages.....	238
Segregation of duties messages.....	239
Rollback messages	239
Release package and sandbox messages.....	240
Settings approval messages	241
Server and license messages	242
Request content messages	242
Execution log notes	243
26.2 Common scenarios.....	243
Cannot sign in.....	243
Cannot connect to a server.....	243
A request does not run	244
The worker does not run.....	244
A report comes back empty	244
Email is not delivered.....	245
Section 27. Governance Controls and Transparency	246
27.1 Missing governance step indicator	246
27.2 Mandatory sandbox trial.....	246

27.3 Integrity of the rollback plan.....	247
27.4 Blocked segregation of duties attempts	248
27.5 Emergency change review	248
27.6 Effect of unmasked column approval on the risk band	250
27.7 Password change on first login	250
27.8 Governance readiness screen	250
The columns of a row.....	251
What a dash in the Expected column means.....	251
The three states and what you need to do	251
The counters at the top of the screen	252
Example: the ticket number format row.....	252
Every row on the screen and what it means.....	253
DAM and SIEM matching declarations.....	260
27.9 Scope and boundary of the rule engine	261
27.10 Disabled buttons and their reasons	261
27.11 Copying a request	262
Section 28. Appendices	263
28.1 Enum glossary	263
Roles (enmRole)	263
Request status (ChangeRequestStatus)	264
Request type (QueryType)	264
Database type (DatabaseType)	265
Risk band (RiskTier)	265
Release package execution mode (ReleaseExecutionMode).....	265
Approval step status.....	265
Server environment	266
Masking regime.....	266
Masking policy on the record.....	266
Masking reason types	266

Candidate column status	267
Settings change status.....	267
License status	267
28.2 Parameter quick reference.....	268
28.3 Permission key quick reference	270
28.4 UTC storage note.....	271
28.5 Installation checklist.....	272
28.6 Initial configuration checklist	272
28.7 Audit preparation checklist	273

Section 0. Introduction and Terms

0.1 What SQL Change Guard is

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

In most organisations change management is already in place: the work is recorded in a ticketing system, discussed by a change advisory board where required, approved there, and application side changes go through a deployment pipeline. SQL Change Guard does not replace that arrangement, it connects to it: the number of the ITSM record is linked to the request, verified in ITSM when the request is saved and again right before execution, and the work done is written to the ITSM record as a note (Section 22).

All of that operates on the record that describes the change. The SQL text that will actually run stays outside the chain, and the link breaks in three places:

1. What the script does is not systematically analysed. Even where the script is attached to the ticket, it is read by eye; which object is touched, whether an unfiltered update is present, or whether a critical table is involved depends on the attention of the reader, and what was actually checked does not appear in the record that remains.
2. There is usually no record showing that the approved text and the executed text are the same text. A one character correction made to the script after approval is invisible to the ticketing system.
3. Reading data out of production stays outside the chain. Even where the change process is mature, a request for a data extract is often handled without a ticket; who took which columns for what reason, and where the result was delivered, is not collected in one place.

SQL Change Guard works on these three points. The script is parsed with the grammar of the database itself rather than treated as text; what is done to which object is derived from the script, a risk band is calculated from the rule set and the required approval steps are created (Sections 9, 10, 12). The approved script is executed by the application and its fingerprint is stored, so it can be shown that the approved text and the executed text are the same (Section 11). Reading data from production is a separate request type: sensitive columns are masked, the result is not displayed on screen and it is delivered as an encrypted package (Section 13). Every step is written to an append-only audit trail on which the database engine rejects updates and deletes (Section 17), and the full life cycle of a single request can be exported as a signed evidence file (Section 18).

The boundaries of the product should be known from the start. The connection to ITSM is limited to reading and writing notes: the product does not open records in ITSM and does not change their state (Section 22). The product does not monitor live database traffic, so it cannot report a change made outside the product on its own; what it can show is that the definition on the server does not match the record (Section 16.3). The rule engine only states what can be read from the script; checks that require the current state of the target database are out of

scope (Section 27.9). For the same reason there is no object dependency analysis: the objects a script touches are listed, but which objects call a procedure being changed is not shown.

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
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

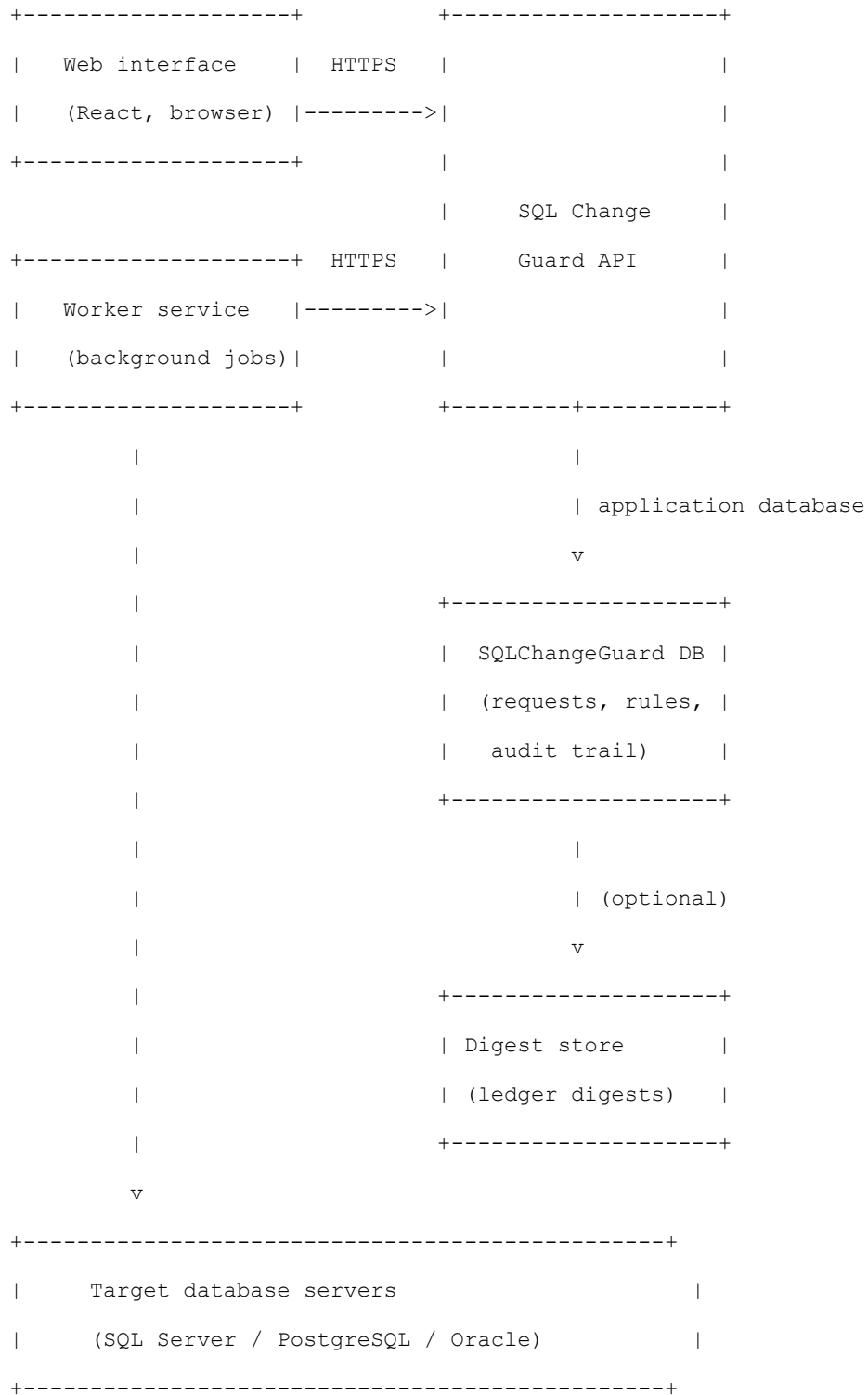
Term	Description
Change Request	A record opened so that one or more SQL scripts run on one or more target servers. Abbreviated CR.
Query Result	A request type opened to read data from a live environment. The result is delivered as an encrypted file. Abbreviated QR.

Term	Description
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 (internal value, not shown)	A numeric value computed during internal evaluation. It is not shown on any screen, in any report or in the evidence dossier. The only measure that matters to users and auditors is the risk band, and the band drives the decision.
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.
Emergency request	A request where the normal approval path is shortened and a written reason is mandatory. It is not a separate request type: it is an emergency flag on the request, and such a request matches only an emergency policy.
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 record of events in the system. The database engine rejects updates and deletes on these records.
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, ledger digest and evidence write boundary checks, maintenance jobs.	No request runs at all. The Run button hands the work to the worker queue and the worker performs the execution, so manually started requests also wait and stay in the Executing state. Email is not sent and server connection status is not refreshed.
Application database	Requests, rules, users, audit trail.	The system stops completely.
External digest store	The folder where the API writes the ledger digests (signed when signing is on) (Evidence:Ledger:DigestOutputPath). It must sit where the database administrator cannot change it; the organisation provides this store (a WORM share, for example).	The ledger keeps working, but no record is covered by an external digest and there is no basis for independent verification.
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: always SQL Server. There is no option.

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
TableWriteLogCoverageJob	Writes the scope summary of the database level write log (row count, id range) to the evidence ledger.	Workers:TableWriteLogCoverageSeconds	86400 seconds (once a day)
HeartbeatJob	Reports to the API that the worker is alive.	Fixed period in code	-
LedgerDigestJob	Starts the ledger digest round: the API takes the digest, signs it (when signing is on), writes it to the digest folder (Evidence:Ledger:DigestOutputPath) and checks for restores and forks (return to an older backup).	Evidence:Ledger:DigestIntervalMinutes	60 minutes
EvidenceBoundaryJob	Checks the evidence write boundary; if it is broken, state changing operations stop.	Evidence:Ledger:BoundaryCheckMinutes	15 minutes

Enable and disable flags:

Flag	Effect
Workers:MainWorkEnabled	If false, request execution stops entirely. Neither scheduled requests nor requests started manually from the screen will run, because the worker performs both.
Workers:ConnectionCheckEnabled	If false, server connection status is not updated.
Workers:MaintenanceEnabled	If false, maintenance tasks do not run.
Workers:TableWriteLogCoverageEnabled	If false, the daily scope summary of the write log is not written to the ledger.

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 and are applied per IP address. With two instances the effective limit for the same address 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
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.

Component	Minimum	Note
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
100	about 2 GB

This is raw growth. The maintenance tasks shipped with the installation only delete expired tokens, notification log records older than 90 days and query result files; they do not archive requests or audit records. Audit records sit in a ledger table and cannot be deleted.

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

A save writes the rows of the request and the audit event in a single transaction. As the number of users saving at the same moment grows, lock waits in the database grow and 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

People saving at exactly the same moment	Attempted	Errors returned to the user
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. The value was measured on a single machine and is measured again on the organisation's hardware during installation acceptance. Running more than one copy of the API does not increase save speed, because the limiting 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 the defined active servers (by default at most eight at a time); a server it cannot reach occupies one of these slots until the connection timeout expires.

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	Database write locks
System wide save ceiling	about 20 requests per second	Database write locks
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; running reports over wide date ranges outside busy hours is recommended.

Section 2. Installation

This section covers a from-scratch installation. The order matters. What to prepare before installation, the IIS and Windows service installation, a database installed by hand by the database administrator and the detailed setup of the integrations (LDAP, SSO, ITSM, SIEM) are in two separate documents: the Installation Prerequisites Checklist and the Installation and Operations Guide.

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 2022 or later	For the application database. Audit records are kept in the append-only ledger table that comes with SQL Server 2022; the ledger exists in every edition.
Windows server	The worker runs as a Windows service. Evidence dossier PDF generation also uses Windows fonts.
Network access	Both the API server and the Worker server must reach the target database servers: the Worker runs requests and the connection health check, while the connection test and the column preview go through the API. 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_SeedData.sql	Roles, default administrator, parameters, SQL standards, report definitions, maintenance tasks
003_StoredProcedures.sql	Report stored procedures
004_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, the product does not touch the schema: the database administrator applies these scripts and then runs the registration file produced by the DbScripts\DbSchemaVersions.ps1 helper in the API folder. If the API finds a missing or changed script at start-up, it names it and does not start. Organisations with strict change control usually prefer this. The steps are in section 3.2 of the Installation and Operations Guide.

2.4 appsettings.json field by field

The table below describes every section of the API appsettings.json. Secrets are not written into this file; how they are supplied is explained just below.

Where secrets are read from: the product reads confidential values such as connection strings and encryption keys from environment variables. Their entries in the configuration file are left empty. Configuration is read in this order and each step overrides the previous one: appsettings.json, appsettings.<Environment>.json, environment variables, command line. The name of the environment variable is derived by replacing the colon in the setting path with a double underscore. For example, Jwt:SigningKey comes from Jwt__SigningKey, and Security:ServerPasswordKey comes from Security__ServerPasswordKey.

Why they are not written into the file: in most organisations the configuration file lives in source control and ends up in backups. A secret sitting there is available to everyone with access to the repository. The heaviest consequence is the evidence signing private key: whoever knows it can produce signed evidence files in the name of the organisation.

The five required secrets. The product does not run without these: ConnectionStrings:Default (the application's own database), Jwt:SigningKey (session token signing key), Security:ServerPasswordKey (encryption of server passwords), Security:QrZipPasswordKey (encryption of the query result package), Workers:ApiKey (how the worker service proves its identity to the API).

The conditional secrets. These are needed only if the related feature is used: Mail:Password (notification email), Ldap:BindPassword (directory authentication), Evidence:Signing:Keys (the key that signs evidence files, packages and the ledger digest), Evidence:Timestamp:ProxyPassword (only if the timestamp authority is reached through an authenticating proxy), Sso:Providers:N:ClientSecret (the OIDC client secret for corporate sign-in) and Sso:Providers:N:SpCertificatePassword (the password of the SAML signing certificate). N is the position of the identity provider in the list; the first provider is 0. Example: Sso__Providers__0__ClientSecret. An undefined secret for an unused feature is not a gap. If the evidence signing key is not defined the product keeps working and evidence files are produced unsigned; the gap is visible both in the file itself and on the Governance Readiness screen.

Two points about the evidence signing key. First: a single key pair is used across the product, and the private key stays only on the API. The worker has the API sign chain seals, and the API takes and signs the ledger digests itself; the private key is removed from the configuration the worker receives from the API. Second: keys are held in a ring (under Evidence:Signing:Keys, each with an id). Signing uses only the key named by Evidence:Signing:ActiveKeyId, while verification can use any key in the ring. When a key is replaced, the public part of the old one must be left in the ring; the verifiability of past evidence files depends on it.

Because the worker fetches its configuration through the API, that call carries the database connection strings (when sharing is on), the Security keys, and the mail and SIEM settings; it does not carry the evidence signing private key. If the API and the worker are on separate machines, the connection between them must be encrypted. The product cannot enforce this; it is a firewall or reverse proxy decision. However, the Governance Readiness screen raises a warning if it detects that the call arrived unencrypted from a remote address.

Example (Windows, persistent system variable). In an elevated command prompt: setx /M ConnectionStrings__Default "Server=SQLSRV01;Database=SQLChangeGuard;UserId=scg;Password=...;TrustServerCertificate=True" and setx /M Jwt__SigningKey "a random value of at least 32 characters". After defining the variables the API and worker services must be restarted; configuration is only read when the process starts. To remove a variable do not use setx, which writes an empty value: use reg delete "HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Environment" /F /V Jwt__SigningKey.

Verification: the Secret handling group on Settings > Diagnostics > Governance Readiness shows, for each of the secrets above, where the value comes from. The secret of every identity provider defined for corporate sign-in is a separate row. There are three states: environment variable, configuration file, not set. The expected value is environment variable. The screen never shows the secrets themselves, only where they come from.

2.4.1 ConnectionStrings

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

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 Database:EvidenceLedger and Evidence:Ledger

Evidence ledger and ledger digest settings. How the ledger works is described in Sections 5.4, 17.16 and 17.17.

Key	What it does
Evidence:Ledger:DigestOutputPath	The folder where the API writes the ledger digests (signed when signing is on). It must be the immutable store of the organisation: a WORM share, storage with object lock, or a share the database administrator cannot write to. An existing file is never overwritten. If left empty, no record is covered by an external digest.
Evidence:Ledger:DigestIntervalMinutes	How often the digest is taken, in minutes. Default 60, at least 5.
Evidence:Ledger:DigestStaleAfterIntervals	After how many intervals without a digest the Governance Readiness screen warns. Default 2.
Evidence:Ledger:BoundaryCheckMinutes	How often the evidence write boundary is checked, in minutes. Default 15.
Database:EvidenceLedger	Creates the evidence table as an append-only ledger. Default true; false is only for integration tests. If set to false in production, the readiness screen warns and the change is recorded as App.ControlRelaxed.

The API application pool account must be able to create new files in the digest folder and read them only; it gets no permission to change or delete files. The Worker account needs no permission on this folder. The database administrator must not have write access to this folder. See Section 17.17.

2.4.3 Auth

Key	What it does	Default	Example effect
Auth:Mode	How a person with no account in the product is verified when they try the password screen. Local: only accounts opened in the product can sign in. Ldap: the person is checked against the corporate directory and, if verified, an account is opened. The account gets the role mapped to the directory group. If no group is mapped, with Ldap:RoleSource Manual the account is opened with the NoAccess role and an administrator assigns the role; with Directory no account is opened. For a person who already has an account this setting is not used; the account's own authentication type (Local, LDAP, Corporate account) applies. New people arriving with a corporate account (SSO) are governed by Sso:AutoProvision, not by this setting.	Local	
Auth:AccountLockoutMaxAttempts	How many wrong password or two-step verification code attempts lock the account. Both count towards the same counter.	5	With 5, the fifth 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	

Key	What it does	Default	Example effect
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.
Auth:MaxSessionHours	The longest a session can last from the first sign-in, in hours. Token renewal does not extend it; when it ends the user signs in again. 0 means no limit.	12	With 12, a user who signed in at 08:00 signs in again by 20:00 at the latest.
Auth:MfaTicketMinutes	How long the two-step verification code can be entered after the password is verified, in minutes.	5	With 5, if the code is not entered within 5 minutes the user goes back to the password step.
Auth:MfaTicketMaxAttempts	How many wrong codes can be tried in a single sign-in.	5	With 5, after the fifth wrong code the user goes back to the password step.
Auth:LocalLoginAllowedUsers	The local accounts that may sign in on the password screen. An empty list means every local account may sign in. When the list is filled, only the listed local accounts sign in with a password; the others are told "Password sign-in is	empty (all local accounts)	With ["admin"] only admin signs in with a password. While corporate sign-in is on, every sign-in of this account is written to the audit trail as a LoginBreakGlass event

Key	What it does	Default	Example effect
	not allowed for this account" and the password is not even checked. While corporate sign-in is on, only break-glass accounts are written here. LDAP and corporate accounts are not affected by this list.		and reaches the SOC as an alert.

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. Renewal does not extend the total length of a session; the upper limit is Auth:MaxSessionHours.	48
Jwt:SigningKey	Signing key of the token. Must be changed in production.	(empty in the file; supplied via the Jwt__SigningKey environment variable)

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.

Total length of a session: when the access token expires the application gets a new one with the refresh token. Renewal does not extend the session: the moment the session started is carried to the new token, and once Auth:MaxSessionHours has passed since the first sign-in, renewal is

refused and the user signs in again. An account closed in the directory therefore loses its open session within that time at the latest. Neither the access token nor the refresh token goes beyond this cap: on the last renewal close to the cap both expiry times are cut to the end of the session, and the expiry date in the session record shows the real end of the session. Even if a browser tab stays open for a long time, the page is reloaded or a new tab is opened, the session continues as long as the refresh token is valid. Tabs share the same session; a new token taken in one tab passes to the others, and signing out in one tab closes all of them. Every refused renewal is written to the audit trail as a `SessionRefreshRejected` event with its reason: `Revoked` (a used token presented again), `Expired`, `SessionLifetimeExceeded`, `NoAccess` or `SsoDisabled`.

On sign-out the refresh token is revoked on the server and the sign-out is written to the audit record under the user's name. Refresh tokens, password reset links and two-step verification tickets are stored in the database as a one-way hash (SHA-256), not as plain text; someone who can read the database cannot sign in with them.

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.

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)
```

These keys must be supplied as environment variables rather than left in plain text inside `appsettings.json`; user secrets are read only in the development environment.

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.

2.4.6 License

Key	What it does	Default
License:LicensePath	Path of the license file.	license.scglicense

Key	What it does	Default
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

Password verification against the corporate directory (LDAP or Active Directory). The person types the corporate username and password on the sign-in page; the product does not store the password, it asks the directory. If Ldap:Host is empty the directory is not used. An invalid value is never used silently: a safe default is used and the error is shown on the Governance Readiness screen; if an error makes sign-in impossible (empty BaseDn, missing domain information or service account required by the bind format, no {user} in the user filter) the directory is not contacted at all.

Key	What it does
Ldap:Host	Name of the directory server, for example dc01.company.local. If empty, the directory is not used.
Ldap:Port	Port. If not set: 636 for Ldaps, 389 for StartTls and None.
Ldap:Security	Encryption of the connection. Ldaps (default, recommended): encrypted from the start. StartTls: connects to the plain port and switches to encryption. None: unencrypted; passwords travel over the network in clear text, only deliberately and for testing.
Ldap:BaseDn	Directory path where people are searched, for example DC=company,DC=local. A person outside this path is not found. Required.
Ldap:BindDn, Ldap:BindPassword	Full name and password of a read-only service account. Required for the SearchDn format and for directory synchronisation. The password is a secret and is given in production through the Ldap__BindPassword environment variable.
Ldap:Domain	Active Directory domain name, for example COMPANY. Required for the DownLevel format. If the person types a different domain (OTHER\ali) the password is not tried at all.
Ldap:TimeoutSeconds	Timeout of a directory call in seconds, between 2 and 60. Default 10.
Ldap:AllowUntrustedCertificate	When true, the certificate of the directory server is not validated. Only for a test environment with a self-signed certificate; it must be false in production, otherwise someone in the middle can pretend to be the directory and collect passwords.
Ldap:BindFormat	Which name is used to verify the password with the directory. DownLevel: COMPANY\user (Active Directory, needs Ldap:Domain). Upn: user@company.local (Active Directory, needs Ldap:UpnSuffix). SearchDn: the person is first searched with the service account, then verified with the full name found (generic directories such as OpenLDAP).
Ldap:UpnSuffix	Active Directory UPN suffix, for example company.local. Required for the Upn format.
Ldap:UserFilter	Search filter that finds the person. {user} is replaced by the typed short name; special characters are escaped. It must return exactly one person. Default: (&(objectClass=user)(sAMAccountName={user})). OpenLDAP example: (&(objectClass=inetOrgPerson)(uid={user})).
Ldap:IdentityAttribute	Attribute holding the permanent identity of the person. Active Directory: objectGUID (default). OpenLDAP:

Key	What it does
	entryUUID. The account is linked to this value at the first sign-in; even if the username changes or a new person gets the same name, the account never moves to the wrong person.
Ldap:EmailAttribute, Ldap:DisplayNameAttribute	E-mail (default mail) and display name (default displayName) attributes. No account is opened for a person without an e-mail address in the directory.
Ldap:GroupMatch	Which name of the group is compared in role mapping. Dn (default, recommended): the full name; a group with the same name in another folder does not match. Cn: the short name; groups with the same name in different folders are not told apart.
Ldap:NestedGroups	When true (default), nested groups count too: if group A is inside group B, a member of A is also a member of B. At most 10 levels and 500 groups are read.
Ldap:RoleSource	Source of the role of a directory account; the same rule as Sso:RoleSource. Manual (default): the directory group grants the role only at the first sign-in; a person with a mapped group is opened with that role, a person without one with no access (NoAccess). After that the administrator manages the role on the Users screen and a group change in the directory does not change it. Directory: the role is calculated from the directory group at every sign-in and synchronisation; no account is opened for a person without a mapped group and the role cannot be changed by hand (except SystemManager). In both modes, if the groups of a new person cannot be read from the directory, no account is opened; the person tries again.
Ldap:Sync:Enabled	When true, the directory is synchronised at regular intervals: the Worker tells the API that sync is due and the API performs it (the Worker never connects to the directory). An account missing or disabled in the directory is left without access (NoAccess) and its sessions are closed; the account is not deleted. Default false; recommended on in production.
Ldap:Sync:IntervalMinutes	Minimum time between two synchronisations in minutes, between 15 and 1440. Default 60.
Ldap:Sync:AbortRatio, Ldap:Sync:AbortMinCount	Safety limit. If the number of accounts to be closed in a synchronisation is not below AbortMinCount (default 5) and the share exceeds AbortRatio (default 0.2, 20 percent), the synchronisation changes nothing and writes the Ldap.SyncAborted event.

Key	What it does
Workers:DirectorySyncPollSeconds	How often the worker asks the API whether synchronisation is due, in seconds. Default 300. The worker does not connect to the directory; directory settings are kept only in the API.

The difference between LDAP users and local users is described in Section 4.8. Users who sign in with a corporate account (SSO) are covered in the same section; the connection settings are in Section 2.4.11.

Trace of setting changes: this section and Auth:Mode live in the configuration file. At every start the API compares a summary of these settings without secrets with the last record; if they differ, the old and new state are written to the audit trail as the Ldap.ConfigurationChanged event and sent to the SOC as an alert.

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
Mfa:EnforceEnrollment	Whether enrolling is mandatory. When off, enrolling is the user's own choice. When on, a user who has not enrolled can sign in but the session is good for the setup step alone.	false
ForwardedHeaders:KnownProxies	IP addresses of the reverse proxy (load balancer) if the product runs behind one. Only the X-Forwarded-For header coming from these addresses is	empty (header is not read)

Key	What it does	Default
	processed, and the real client IP address is written to the audit record.	

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. So the master switch being on does not mean everyone in the organisation is using two-step verification; how many users have set it up is shown on the MFA enrollment coverage row in Settings > Diagnostics.

Directory (LDAP) accounts can set up two-step verification as well. The directory verifies the password but offers no second factor; the product asks for the code after the directory password is verified. When enrollment is mandatory, LDAP accounts go through the setup step too.

Making enrollment mandatory: when Mfa:EnforceEnrollment is switched on, a user who has not yet set up two-step verification can still sign in, but that session is good for the setup step alone. Once the password is verified the setup step opens directly on screen: a QR code, a key that can be typed in by hand and a six digit confirmation field. Nothing else runs until the code is confirmed; the rule is enforced on the server rather than in the interface, so requests are rejected even if another screen is reached through the address bar. As soon as the code is confirmed the session returns to normal and there is no need to sign in again. This setting only means something while the master switch is on. With the master switch off the enforcement flag does nothing on its own; otherwise users would be sent to a setup they cannot complete and would be locked out. Whether the setting is on is shown on the MFA enrollment enforced row in Settings > Diagnostics. Security of the code step: only a person who has verified their password (or directory password) can reach the code screen. Once the password is verified the server issues a short-lived, single-use	Folder where query result ZIP files are written.	C:\SCG\QrFiles
--	---	-----------------------

ticket for that sign-in (Auth:MfaTicketMinutes); the code is accepted only together with this ticket. Knowing a user name and a code is not enough to sign in. At most Auth:MfaTicketMaxAttempts wrong codes can be tried with one ticket; then the user goes back to the password step. Wrong codes count towards the same lockout counter as wrong passwords and can lock the account. A code that has been accepted is not accepted a second time, even while it is still valid. Every successful and failed code attempt is written to the audit record with the real user name and the IP address of the connection (MfaSuccess, MfaFailed). Running behind a reverse proxy: the IP address in audit records is read from the connection itself; the X-Forwarded-For header sent by the client is not trusted, because anyone can write that header. If the product runs behind a reverse proxy or load balancer, the IP address of the proxy is listed in ForwardedHeaders:KnownProxies. Only the header coming from proxies in this list is processed and the record gets the real client		
---	--	--

address. If the list is empty, every record shows the address of the proxy. QueryResult:OutputPath		
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.	(empty in the file; supplied via the Workers__ApiKey environment variable)
Workers:WorkerErrorMailTo	Address that receives notifications when the worker hits an unexpected error.	
Workers:StaleExecutionRecoveryMinutes	Crash recovery threshold in minutes. An execution claimed for longer than this is presumed dead. A request that certainly did not commit on the target returns to Approved; a request that may have committed moves to ReconciliationRequired and is not run again. 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.

2.4.10 SIEM (SOC integration)

The product can forward governance events to the organisation's SOC collector. This section holds the connection details: which address, which protocol and which format. What gets sent is not managed here but from the parameters on the Settings screen (Section 7.11).

The split is deliberate. Address, port and certificate are installation details; the person who installs the product knows them and they rarely change. Scope is a control: it decides what reaches the SOC, its change must appear in the audit trail and it must be subject to the four

eyes rule. If both lived in the same place, narrowing the scope would only require access to a file, and that change would be visible nowhere.

This section is not written separately on the worker side. The worker receives the same settings through the API. If they were kept in two places, the API and the worker could write to different endpoints and nothing would report it.

Key	What it does	Value shipped with the installation
Siem:Endpoints[].Name	Endpoint name. The delivery marker is kept under this name. If it is changed after installation the endpoint counts as new, its marker restarts from zero and every past event is sent again.	SOC
Siem:Endpoints[].Type	Syslog is the real target. File writes one event per line to a file; for organisations without a collector and for trying out the installation.	File
Siem:Endpoints[].Host	Address of the collector.	127.0.0.1
Siem:Endpoints[].Port	Port of the collector. 6514 is the common value for encrypted syslog.	6514
Siem:Endpoints[].Protocol	Tls, Tcp or Udp. Tls is recommended. UDP loses a message that does not fit into a single packet and does not report the loss; use it only if the organisation insists.	Tls
Siem:Endpoints[].Format	Rfc5424Json, Rfc5424Sd, Cef, Leef or Json. Chosen to match the collector: Cef for	Json

Key	What it does	Value shipped with the installation
	ArcSight, Leef for QRadar, Rfc5424Json for most others.	
Siem:Endpoints[].Facility	Syslog facility name. Used at the collector to tell sources apart.	local0
Siem:Endpoints[].TlsServerCertificateThumbprint	If left empty the system trust chain is used. If filled in, the collector certificate is compared against this thumbprint and the connection is refused when it does not match. Fill it in for collectors that use a self signed certificate.	(empty)
Siem:Endpoints[].FilePath	Used only when Type is File. Path of the file.	logs/siem-events.jsonl
Siem:Endpoints[].BatchSize	How many events are sent at most in one round. If the collector stays down for a long time, the accumulated records flow in groups of this size.	200
Siem:Endpoints[].PollSeconds	Stored on the endpoint definition but it does not drive the schedule. The delivery interval comes from the worker side key Workers:SiemPumpSeconds, and all endpoints are served in the same round.	10
Siem:Endpoints[].HeartbeatMinutes	Stored on the endpoint definition but it does not	5

Key	What it does	Value shipped with the installation
	drive the schedule. The heartbeat interval is five minutes across the product.	
Siem:AppLogQueueSize	Capacity of the in memory queue for the product's own log. When it fills up the oldest record is dropped and a counter increases.	10000
Siem:AppLogMinLevel	Product log below this level is not sent to the SOC. Sending Information floods the SOC with the product's internal detail.	Warning
Workers:SiemPumpSeconds	Delivery poll interval in seconds. With the default value an audit record reaches the SOC within ten seconds at most. Zero or negative values are ignored and 10 is used; the upper bound is 3600 seconds.	10
Workers:SiemLeaseSeconds	Duration of the single writer lease per endpoint and per stream, in seconds. If more than one worker copy runs, the second copy will not write to the same stream before this period expires.	60

More than one endpoint can be defined. Each endpoint keeps its own delivery marker; when one becomes unreachable the other is unaffected.

The installation ships with the value File, so that delivery can be seen working right after installation without a collector in place. When connecting to the organisation's collector, Type is set to Syslog and the address, port, protocol and format are filled in.

2.4.11 Sso (corporate sign-in)

When corporate sign-in (single sign-on, SSO) is on, users enter their password and multi-factor verification on the organisation's own sign-in page; the password never reaches the product. Two standard protocols are supported: OIDC (Microsoft Entra ID, Keycloak, Okta) and SAML 2.0 (AD FS, Entra ID, Keycloak). The verified person is found or created by their permanent identity, their group is mapped to a role and the product starts its own session. It is off by default. Restart the API after a change.

Key	What it does	Default
Sso:Enabled	When true, the sign-in screen shows a "Sign in with ..." button for every identity provider. When false, corporate accounts cannot sign in, and once the API is restarted their open sessions are closed as well (for example when the security of the identity provider is in doubt); local and LDAP sign-in are not affected.	false
Sso:ApiBaseUrl	The address of the API as seen by the browser. The return address given to the identity provider is built from it. It is not derived from the request, so the right address is produced behind a reverse proxy too.	http://localhost:5204
Sso:FrontendBaseUrl	The address of the web interface. After sign-in the browser returns here. The return address is read only from this setting, so an	http://localhost:5173

Key	What it does	Default
	attacker cannot supply a redirect address.	
Sso:AutoProvision	When true, a person with no account is created at their first sign-in. When false, an administrator first opens the account on the Users screen with the Corporate account type and the same user name.	true
Sso:RoleSource	Idp: the role is recalculated from the identity provider group at every sign-in; a person removed from the group loses the role at the next sign-in and their open sessions are closed. Manual: the group decides the role only when the account is first opened; after that the administrator manages it.	Idp
Sso:MaxSessionHours	The longest duration of a session opened with a corporate account (hours). The smaller of this and Auth:MaxSessionHours applies. The product session of an account closed at the identity provider ends within this time at the latest.	10
Sso:SignOutFromIdp	When true, a user signing out of the product is also signed out of the identity provider (OIDC only).	false

Key	What it does	Default
Sso:LoginTimeoutMinutes	The time allowed to come back from the identity provider after the sign-in button is pressed (minutes).	10
Sso:ExchangeCodeSeconds	The lifetime of the one-time code that the interface exchanges for a session after the return (seconds). The code is valid only in the browser that started the sign-in.	120
Sso:ClockSkewMinutes	The clock difference accepted when checking the times in the identity provider response (minutes).	5
Sso:Providers	The list of identity providers. Each one is a separate button on the sign-in screen. Its fields are in the table below.	empty

Fields of each identity provider:

Key	What it does
Name	Permanent name. Linked identities and role mappings are stored under this name; do not change it after installation. Letters, digits, hyphen and underscore only.
DisplayName	The name shown on the sign-in button.
Protocol	Oidc or Saml.
RequireHttpsMetadata	Whether the metadata address must be HTTPS. Always true in production.
Authority, ClientId, ClientSecret, Scopes	For OIDC: the provider address, the client id given at app registration and the client secret.

Key	What it does
	The client secret must be supplied through an environment variable.
IdpMetadataUrl, SpEntityId, SpCertificatePath, SpCertificatePassword, SignAuthnRequests	For SAML: the identity provider metadata address, the product's SAML identity and an optional signing certificate. The certificate password must be supplied through an environment variable.
SubjectClaim	The permanent identity that matches the person. Entra ID: oid, Keycloak and Okta: sub, SAML: NameID (must be persistent). Never email or user name, because those can change.
UsernameClaim, EmailClaim	User name and email of a newly opened account.
GroupsClaim	The information read for role mapping. With Entra ID, app roles (roles) are recommended.
RequireMfaClaim, MfaClaim, MfaClaimValue	When true, sign-in is rejected if the identity provider does not state that multi-factor verification was done. Recommended as true in production.

The addresses to enter at the identity provider (return address, post sign-out address, SAML identity and metadata address) are shown ready to copy on Settings > Corporate Identity (SSO and LDAP). The Test connection button on the same screen shows whether the identity provider metadata can be reached. A provider defined with errors is not used and appears on the Governance Readiness screen in the Corporate sign-in configuration errors row.

Which group gets which role is not set in this file but in the Group to role mapping table on Settings > Corporate Identity (SSO and LDAP) (Section 4.8).

Trace of setting changes: this section lives in the configuration file, so it cannot be changed from a screen. A change made in the file (turning corporate sign-in off, removing the MFA check, changing the role source, changing the Auth:LocalLoginAllowedUsers list and similar) still leaves a trace: at every start the API compares a summary of these settings with the last record and, if they differ, writes the old and new state to the audit trail as the Sso.ConfigurationChanged event. The event reaches the SOC as a control change and appears in the User Access Changes

report. Secrets are never written into the summary, only whether they are set. Restarting with the same settings produces no record.

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 cannot fetch its configuration from the API at startup; after 10 attempts 3 seconds apart it stops and no job runs. It appears offline in the interface. If the worker is to connect to the application database with its own account, ConnectionStrings:Worker (or the ConnectionStrings__Worker environment variable) is set in this file; when it is empty the connection string comes from the API.

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)

The system forces this password to be changed at first sign-in: until it is changed the session only allows the password change operation and no other request works. The check is server side and applies even if the interface is bypassed. This account ships alone in the production copy; no other sample user is created.

Local accounts (admin included) sign in with the user name only: admin. With a domain or e-mail suffix, such as COMPANY\admin or admin@company.com, the password is not tried and the message "sign in with the user name only" is shown; this attempt is not counted as a failed

sign-in and does not lock the account. Domains and suffixes are used only by directory (LDAP) accounts, where they are checked against the Ldap:Domain and Ldap:UpnSuffix settings.

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.
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. Only active servers are counted; if the license marks non-production servers as free (the default), only active servers in the Production environment are counted. See 3.4.	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.	-

The product is sold in a single form: every module is enabled. The price depends on the number of production database servers in scope; modules are not bought separately and the number of users is unlimited. The module list in the licence file is a technical field: leaving it empty enables every module, which is what a standard installation does.

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.
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 servers that count towards the limit. Only active servers are counted; inactive servers are not. If the licence marks non-production servers as free, only active servers in the production environment are counted and test, development and staging environments are unlimited. When the limit is reached no new server can be added; existing servers keep working.

Example: with MaxInstances set to 3 and three counted servers, adding a fourth fails. Deactivating a server does free a slot; deleting it is not required. An inactive server does not count towards the limit, but it cannot be used either: it cannot be selected on new requests.

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.
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. It is the highest approval level on the technical side; it is not the organisation's change advisory board (CAB).
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.

ChangeManager and the organisation's change advisory board (CAB) are not the same thing. The CAB decides, in the organisation's ITSM system (such as ServiceNow or Jira), whether a change will be made and when. ChangeManager in the product approves the script and the data that carry out that change: it is used in approval policy steps defined as "approved by at least this role" and in some of the product's permission thresholds. Example: the lowest role

that may download another person's query result file (QrDownloadMinimumRole) and the role that approves the e-mail address a result is sent to (QrEmailApproverRole) are ChangeManager at installation.

Give this role to the person who has the final technical say, not to the person who runs the CAB; in most organisations this is the DBA team lead. If the organisation wishes, the person who runs the CAB can also be given this role, so that on high risk work they see and approve the script as well. This is not a requirement.

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.

Permission comparison looks at the role's level in the hierarchy, not at its id number. Because of this separation, a role added to the list later does not gain authority on its own simply because its number is higher.

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
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

Permission	NoAccess	Viewer	Auditor	Requester	RequestManager	DbAdmin	ChangeManager	SystemManager
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
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

Key	Action it enables
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. For local and directory (LDAP) accounts the user name is the short name: it cannot contain a domain (COMPANY\ali) or an e-mail suffix (ali@company.com), because the sign-in screen reduces the typed name to the short name before looking it up and an account saved in that form could never sign in. The screen suggests the short name in that case. For a corporate (SSO) account the name stays as the identity system sends it (for example ali@company.com). For the same reason a corporate account whose name contains a domain or a suffix cannot be changed to the local or directory type.
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. The screen warns before saving and shows the e-mail of the closed account: the past requests, approvals and audit records of that

Action	Behaviour
	account will appear as belonging to the revived account. The account is not reopened until the administrator chooses "Same person, reopen"; if it is a different person, another user name is chosen. When four-eyes approval is on, the warning appears before the change is sent for approval. Reopening is written to the audit trail as the User.Reactivate event.
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 wrong passwords or two-step verification codes 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. The session is closed. When a role changes, an account is closed, the authentication type changes, a role permission matrix is updated, or a password is reset (by an administrator or through the e-mail link), the sessions of the affected users are closed at once: an open window returns to the sign-in screen on the next click. A session version kept for each user makes this possible; the session key carries the version current when it was issued, and every request compares it with the database. When users change their own password, all their other sessions are closed and the session they changed it from continues without interruption.

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.

When a role or its permissions change, the user signs in again and continues with the new permissions. Withdrawing and granting a permission therefore both take effect at once; nobody keeps working with a session that carries old permissions.

4.8 LDAP users versus local users

Topic	Local user	LDAP user
Where the password lives	In the SQL Change Guard database, as a BCrypt hash; it cannot be reversed.	In the corporate directory. The application never stores it.
Changing the password	Inside the application.	In the corporate directory.
Role	Assigned inside the application.	With Ldap:RoleSource Manual it comes from the directory group at the first sign-in (no access if no group is mapped) and is assigned inside the application after that. With Directory it comes from the directory group and is recalculated at every sign-in and synchronisation.
Account lock	Managed by the application.	The application also counts wrong directory passwords and locks the account. The application threshold must be kept below the directory threshold.
Two-step verification	Set up inside the application; the code is asked after the password.	Set up inside the application; the code is asked after the directory password.
Opening the account	The administrator opens it on the Users screen.	The administrator opens it, or it is opened at the first sign-in when Auth:Mode is Ldap. No account is opened for a person without an e-mail address in the directory.
Recognising the person	By username.	By the permanent identity in the directory (objectGUID or entryUUID). Even if the username changes the person signs in to their own account; a new person with the same name cannot take over the old account.
Leaving the organisation	The administrator closes the account.	Access of a person disabled or deleted in the directory is closed at directory synchronisation and their open sessions end.

Group to role mapping: mappings of directory accounts are kept in the same table as corporate account mappings, on Settings > Corporate Identity (SSO and LDAP), with the "Directory (LDAP)" provider. The value is the full name of the group in the directory, for example CN=DBA-Team,OU=Groups,DC=company,DC=local; spaces and letter case do not matter. If several groups match, the highest role is given. SystemManager is never granted by a mapping. If you do not know the value, let the person try to sign in once; the LdapLoginDenied record in Audit Logs lists the groups received from the directory (ReceivedGroups).

When the role comes from the directory (Ldap:RoleSource = Directory): if the person is removed from the group, they get the new role at the next sign-in or synchronisation and their open sessions with the old role are closed at once. If the group information cannot be read, an existing role is not lowered and sign-in is rejected. On the Users screen the role can be changed by hand only to or from SystemManager. If a person must be stopped in the product at once, without waiting for the directory team, delete the account on the Users screen: their open sessions are closed at once and they cannot sign in even with the correct directory password. The account can be added again with the same name if needed; past records are kept.

Directory synchronisation: with Ldap:Sync:Enabled on, the worker looks up the directory accounts with the service account. An account that is not found or is disabled in the directory is left without access and its sessions are closed; the e-mail address is updated from the directory. If the mass deactivation safety limit is exceeded or the directory cannot be reached, nothing is changed. The last SystemManager account is never closed. Synchronisation can also be run with the Synchronise now button on Settings > Corporate Identity (SSO and LDAP); the Test connection button on the same screen shows whether the directory can be reached.

Wrong password: for a person with an account, a wrong directory password is counted as for a local account and the account is locked. For a name with no account in the product, the directory is not contacted after the same number of wrong attempts for the same time; existing and unknown names get the same answer. If the directory cannot be reached, the counter does not increase. A SystemManager account is never locked permanently by the product; the same limit is applied in memory to these accounts, so the administrator's directory password cannot be guessed without limit through the product screen.

Audit trail: an account opened by a directory sign-in is written as User.CreatedFromDirectory, a role or e-mail updated from the directory as User.SyncedFromDirectory, an account closed by synchronisation as User.DisabledInDirectory, a rejection by rule as LdapLoginDenied, and the result of a synchronisation as Ldap.SyncCompleted or Ldap.SyncAborted.

Users who sign in with a corporate account (SSO)

When corporate sign-in is on, a "Sign in with ..." button appears below the password fields on the sign-in screen. The user presses it, signs in with their corporate account on the organisation's sign-in page and returns to the product. On the Users screen the authentication type of these accounts is shown as Corporate account.

Topic	Corporate account user
Where the password lives	In the organisation's identity system. The product never sees or stores it.
Changing and resetting the password	In the organisation's identity system. There is no password reset button for these accounts in the product.
Role	Given from the identity provider group according to the Group to role mapping table. When several groups match, the highest role is given.
Two-step verification	Done in the organisation's identity system. It is not set up inside the product, and the product's enrolment requirement does not apply to these accounts.
Closing the account	An account closed in the organisation's identity system cannot sign in again. Its open session ends at the latest after <code>Sso:MaxSessionHours</code> .
Sign-in on the password screen	Not possible. Entering the user name on the password screen returns "This account signs in with the corporate account" and the failed sign-in counter does not increase.

Finding the person: the product finds the person by the permanent identity given by the identity provider, not by email or user name. At the first sign-in the person is linked to an account of the Corporate account type opened earlier with the same user name. They are never linked automatically to a Local or LDAP account; in that case sign-in is rejected and an administrator must change the account type to Corporate account. This way nobody who receives the same name at the identity provider can take over the product's local administrator account.

Group to role mapping: on Settings > Corporate Identity (SSO and LDAP) each row is an identity provider, a group (or app role) value and a role. The value is written as the identity provider sends it; upper and lower case are treated the same. A person who matches no mapping gets no access. If you do not know which value the identity provider sends, let the person try a corporate sign-in once: the sign-in is rejected and the SsoLoginDenied record in Audit Logs lists the group values received (ReceivedGroups). Write that value into the mapping as it is. The SystemManager role is never given by a mapping; it is only given by hand on the Users screen. Changing mappings requires the permission of the Users screen; when settings change approval is on, the change goes through a second person's approval. When the role source is Idp and a mapping is changed or deleted, the open sessions of the people linked to that identity provider who hold the role of the mapping are closed at once; at their next sign-in they get the role given by the new mappings. Nobody can keep working with a permission that came from a deleted mapping.

When the role source is Idp (Sso:RoleSource) the role is recalculated at every sign-in. If the person is removed from the group at the organisation, they get the new role at the next sign-in and their open sessions with the old role are closed at once. In this mode the role of a corporate account can be changed by hand on the Users screen only to or from SystemManager. If the identity provider sends no group information at all, an existing account is not demoted and sign-in is rejected; a new person's account is not opened.

Linked identities: the edit window of a corporate account on the Users screen shows its linked corporate identities (identity provider, permanent identity, last sign-in). The Unlink button removes the link and closes the open sessions. The user is linked again at their next corporate sign-in.

Break-glass account: at least one local, password-protected account with the SystemManager role is kept so that the product can be entered when the identity provider is unreachable. Only these accounts are written to Auth:LocalLoginAllowedUsers; other local accounts cannot use the password screen. Every sign-in of a break-glass account is written to the audit trail as a LoginBreakGlass event and reaches the SOC as an alert.

Using the break-glass account: not every SystemManager is written to this list. Administrators who do the daily administration sign in with their corporate accounts; the SystemManager role is granted to them by hand on the Users screen. Only one or two local SystemManager accounts reserved for emergencies are written to the list. Recommended setup:

Account	Type and role	In the list	Who uses it
breakglass.admin	Local, SystemManager	Yes	Nobody in daily work; only while the identity provider is down
breakglass.admin2	Local, SystemManager	Yes	Spare; if the first account is locked or its password is lost
john.smith	Corporate account, SystemManager (granted by hand)	No	Daily administration
All other users	Corporate account	No	Daily use

With this setup the value in the configuration file is: "Auth": { "LocalLoginAllowedUsers": ["breakglass.admin", "breakglass.admin2"] }. The list lives in the configuration file; a change takes effect when the API is restarted and is written to the audit trail as the Sso.ConfigurationChanged event.

Recommended rules for the break-glass account:

1. The account belongs to a duty, not to a person. Its name is not a person's name (for example breakglass.admin) and it is not used in daily work.
2. The password is long and complex. It is kept in the organisation's safe or in a password vault; it can be split into two halves held by two different people. It is changed after every use.
3. Every sign-in is written to the audit trail as LoginBreakGlass; when SIEM is connected the security team receives an alert. The security team confirms that there really is an outage at that moment.
4. The account is tried at regular intervals (for example every three months) to confirm that it works. The trial is written to the audit trail too, so it serves as evidence in an audit.
5. The number of accounts is kept small. Every local account stays outside the organisation's identity policy (second factor, account disabling).

The admin account that comes with the installation: if it is used as the break-glass account it stays in the list and its password is kept under the rules above. If it is not, new local SystemManager accounts such as breakglass.admin are created and written to the list first; only after that is admin removed from the list. If no local SystemManager is left in the list, no

administrator can sign in while the identity provider is down; the "Break-glass account" row in the Setup Check warns in that case.

Other local accounts: a local account can be added to the list temporarily for people who have no record in the identity provider (for example an external consultant) or during a migration, and removed when the work is done. People who have a corporate account are not given an extra local account "just in case": the person would have two identities, and when they leave the organisation the local account stays open even though the corporate account is disabled. If a person's work must not stop while the identity provider is down, the administrator signs in with the break-glass account and moves that person to a local account temporarily (see When a person cannot sign in with the corporate account or the directory).

Audit trail: a successful sign-in is written as `SsoLoginSuccess`, a technical verification error as `SsoLoginFailed`, and a rejection by rule (account not linked, no role mapped, MFA not stated and similar) as `SsoLoginDenied`. Opening an account from the identity provider is `User.CreatedFromIdp`, updating its role or email is `User.SyncedFromIdp`, identity links are `User.ExternalIdentityLinked` and `User.ExternalIdentityUnlinked`, and mapping changes are `SsoRoleMapping.Create`, `SsoRoleMapping.Update` and `SsoRoleMapping.Delete`.

Session number: every successful login gets a session number. It is written to the login record (`LoginSuccess`, `MfaSuccess`, `SsoLoginSuccess`, `LoginBreakGlass`) and to the identity details of the session: where the password was verified, the identity provider name, MFA and emergency login. These details do not change during the session and are carried over on refresh. With LDAP, even if the person signs in as "KURUM\ayse" or "ayse@kurum.local", the login record is written with the registered account name (ayse). Decisions on a request freeze these details at the moment of the decision (Section 18.8).

Role source: corporate account (SSO) and directory (LDAP) together

`Sso:RoleSource` for corporate accounts and `Ldap:RoleSource` for directory accounts follow the same rule. The group mode (`Idp`, `Directory`) calculates the role from the group at every sign-in. The manual mode (`Manual`) uses the group only when the account is first created; after that the administrator manages the role on the Users screen.

When a person signs in for the first time:

Situation	SSO Idp	LDAP Directory	SSO Manual	LDAP Manual
Has a mapped group	Created with that role, signs in	Same	Created with that role, signs in	Same
Has no mapped group	No account is created	Same	Created with no access (NoAccess); the administrator grants the role	Same
Groups could not be read	No account is created	No account is created; the person tries again	Created with no access	No account is created; the person tries again

After the account exists, in group mode a person removed from the group loses the role at the next sign-in and their open sessions are closed; the role is locked on the Users screen (only SystemManager can be granted and taken back). In manual mode a group change does not change the role and the administrator changes it freely.

Why the two manual modes differ when groups cannot be read: for the directory the product reads the groups itself and recognises a read error with certainty, so it does not create the account; the person tries again and is created with the right role once the directory recovers. For corporate accounts the groups arrive ready from the identity system; if no group information arrives at all, the product cannot tell whether the person has no group or the setup is incomplete, and treats the person as having no group. Neither behaviour grants anyone extra rights. For people in many groups in Entra ID, application roles are recommended instead of groups.

When a person cannot sign in with the corporate account or the directory

First find the reason: in Audit Logs the SsoLoginDenied, SsoLoginFailed or LdapLoginDenied record holds the reason and the groups received.

Situation	Permanent fix	Is a local account a suitable temporary fix?
The identity system or the directory is down	Fix the identity system or the directory; meanwhile	Only for the few people whose work must not stop, temporarily

Situation	Permanent fix	Is a local account a suitable temporary fix?
	administrators sign in with the break-glass account	
The person's group is not mapped to a role (group mode)	Add the person to the right group or map the group	Not recommended: the role would be granted outside the group
Groups cannot be read	Fix the directory problem; use application roles in Entra ID	Temporarily, until the problem is fixed
The person is disabled or locked in the identity system or the directory	This is the organisation's decision; talk to the identity team	No: a local account overrides that decision
Message "an account with this user name exists that is not linked to a corporate account"	Change the account type to Corporate account	Not needed
The corporate account's second factor is lost	Have it reset in the identity system	No
The directory account's product MFA is lost	Users > Reset MFA	Not needed
The account is locked in the product after wrong passwords	The lock expires or the administrator unlocks it	Not needed

To move a corporate or directory account to a local account you do not need to remove the link first: on the Users screen set the authentication type to Local and save (the person's open sessions are closed), then give a temporary password with Reset Password. The person changes the password at the first sign-in and sets up the product's second factor if it is mandatory. If Auth:LocalLoginAllowedUsers is filled, the person cannot sign in with a password until added to the list. The link to the identity system stays on record but is not used; leave it for a temporary move, remove it with Unlink for a permanent one. If the user name contains a domain or an e-mail suffix (for example ayse@company.com) the account cannot be changed to the local type; a separate local account is created for the person with a short name and a different e-mail.

Consequences of a local account: while changing the type the administrator can also change the role, so the role can be granted outside the group. Disabling the person in the identity system or the directory does not close the local account and directory synchronisation does not touch local accounts; if the person leaves, the local account must be closed by hand. The password and the second factor are kept in the product. If a separate local account is created the person has two accounts, their history is split and segregation of duties does not work between the two accounts; use this only when changing the type is not possible and delete the local account when the work is done.

Controls that make this visible: a type and role change goes to approval when four-eyes approval is on and is written to the audit trail as User.Update with old and new values; while corporate sign-in is on, every local sign-in is also written as LoginBreakGlass and raises an alarm when SIEM is on; the "Password sign-in limited to break-glass accounts" row in the Setup Check warns while the allow list is empty; the User Access Changes report lists the change. When the problem is fixed, set the type back to Corporate account or LDAP: the product deletes the local password, the person signs in as before and in group mode the role is calculated from the group again.

A SystemManager can sign in with a corporate account or the directory. The SystemManager role is never granted by a mapping, because adding a member to a group must not create a full administrator of the product; but the administrator can grant SystemManager to a corporate or directory account by hand on the Users screen. Sign-ins and synchronisation do not touch a SystemManager granted by hand. Because nobody can sign in when the identity system or the directory is down, at least one local SystemManager (break-glass account) must always exist; the "Break-glass account" row in the Setup Check verifies this.

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.
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 also used elsewhere in the product: the script fingerprint, the file digests of evidence packages and the digest of the evidence ledger. 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.

Field	What it does	Required	Notes
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.
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.

4.12 Bulk user import

Screen: Settings > Users > Import CSV

Permission: settings.users

The organisation's user list can be loaded from a file instead of being typed in one by one. Import is not a shortcut around adding users individually: the same roles, the same permission check and the same audit trail apply. When the four eyes rule is on, an import also goes for approval and no user is created until it is approved.

Preparing the file

The Download template button gives you an Excel file. Its Users sheet has five columns and the header row must not be changed:

DomainUsername, Email, Role, AuthType, ManagerUsername

The role and authentication type columns are filled from drop-down lists. The role list contains every role the product recognises, including Auditor. Once filled in, save the file with Save As in CSV UTF-8 format and upload it from the screen.

Validation and error reporting

Every row is validated one by one. If even a single row has an error, no user is created and the errors are listed on screen with their row numbers. This is deliberate: a half loaded user list is harder to correct than one that was never loaded.

What is checked: required fields, e-mail format, whether the role and the authentication type are valid, uniqueness of the username and the e-mail both inside the file and in the system, that local and directory account names are short names (no domain and no e-mail suffix), and the absence of a cycle in the manager chain. A username that belongs to a disabled account cannot be created by import either; reviving an account is an administrator decision and is done individually.

The person named in the manager column must be either in the same file or already registered in the system. When users are loaded in batches, people whose manager arrived with an earlier file are linked directly. A closed account cannot be selected as a manager: it would leave users attached to somebody who is not in the list, and the approval step would fall to nobody.

A file can contain at most 500 rows.

An imported account cannot sign in

The template has no password column. This is deliberate: the file carries no password as it is passed around, and no password known to the administrator comes into existence. As a result an imported local account is created without a password and the user cannot sign in.

The user list shows this with a No password badge. To put the account into use, give it a temporary password with Reset password on the row; the user must then set their own password at first sign in. Directory (LDAP) accounts do not get this badge because the product does not hold their password, so they are usable right after the import.

Audit trail

An import writes the same record for each user as adding that user individually would: a User.Create event with the account identity and the full picture of the record. The answer to "who created this account, when, and with which role" therefore does not depend on which way the account was created.

In addition the batch itself is recorded as a User.BulkImport event that names the users it covered. Together the two records show that accounts created in the same minute were not a coincidence.

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.
Making changes to the audit trail detectable	Append-only ledger (SQL Server ledger) and a signed external digest	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. With GCM, tampered ciphertext fails the authentication tag and decryption returns an error.

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 in the append-only ledger table; the database engine rejects updates and deletes.

Data	State
Log files	Password and secret fields are masked.

5.4 Audit trail: append-only ledger

What the ledger is

Audit records are kept in a SQL Server append-only ledger table (evidence.AuditLogs). The database engine rejects changing or deleting a record in this table for everyone, sysadmin included. Dropping the table is possible, but a drop leaves a trace in the ledger's own history and the product catches it (Section 17.16).

The digest and the external witness

The engine links records together in blocks and, on request, produces a single digest that covers the whole ledger up to that moment. Once an hour the Worker starts a digest round; the API takes the digest, signs it with the organisation's private key when signing is on, and writes it to the immutable store of the organisation (Section 17.17). A direct change to the database file or a return to an older backup shows up when compared with a digest written outside.

In every round the API checks the restore history and continuity with the last digest in the external store; if it finds a problem it writes a Ledger.RestoreDetected or Ledger.ForkDetected event. The full independent verification is made by the organisation or the auditor: a backup is restored to a copy and the sys.sp_verify_database_ledger procedure is run with the digests from the immutable store. The steps are in the Auditor Kit document.

Why there is no secret key

The integrity of the records does not depend on a secret key. Everything needed for verification (the digest files, the public key, a copy of the database) can be given to the auditor, and no secret of the organisation is shared. The signature on the digest files only shows which installation a file came from.

5.5 Database level write log

Alongside the audit trail a second record is kept: TableWriteLogs. This record covers the tables in the product's own database; it does not cover the tables in the target databases you manage. 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	Sees changes that bypass the application (while the triggers are enabled)
Sees changes that bypass the application	No	Yes
Seal	Append-only ledger and external digest	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 sits in the append-only ledger: the database engine rejects changes and deletions, and the digest of the ledger is written outside. That is where the audit evidence lives.

The write log is not made a ledger table: it stores the old and new state of a row on every write, so its volume is large and it may need to be cleaned up over time; rows cannot be deleted from a ledger table. Instead, 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, the application account cannot update or delete rows in the write log.

2. Keep the daily scope summary running. Every day the worker writes a scope summary of the write log (row count and id range) to the evidence ledger as a TableWriteLog.CoverageRecorded event. Because the event cannot be changed, bulk deletion becomes visible afterwards. This happens on its own as long as the worker runs (Workers:TableWriteLogCoverageEnabled).

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

Two more fields are not in the table. Validate the server certificate (off by default): when on, the TLS certificate of the target is validated; SQL Server no longer trusts the certificate unconditionally and PostgreSQL performs full validation (VerifyFull). When off the connection may be encrypted but the identity of the target is not verified; turning it on for production targets is recommended. Sandbox required: visible when the Sandbox module is licensed, and makes a	What it does	Required	Valid values	Default	If left empty	Effect of changing it
---	--------------	----------	--------------	---------	---------------	-----------------------

sandbox run mandatory for requests on this server. Field						
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. On SQL Server the connection opens in this database; a USE statement in the script overrides it. For Oracle it is the service name, usually supplied	Depends on type	Free text	Empty	The default database is used on SQL Server	

	through the host.					
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	-	On / Off	Off	-	When on, requests without a rollback

	rollback script.					cannot be executed.
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.

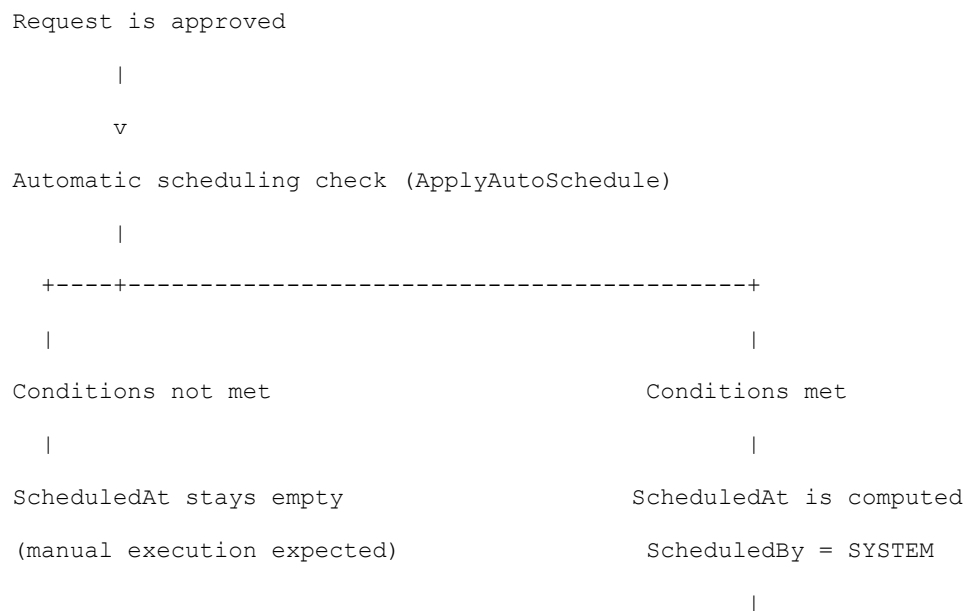
Result	Meaning	What to do
Success	The connection was established.	-
Authentication error	Username or password is wrong.	Check the credentials.
Server unreachable	Network, host name or port is wrong, or a firewall is blocking.	Check network access.
Database not found	The database name is wrong.	Check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

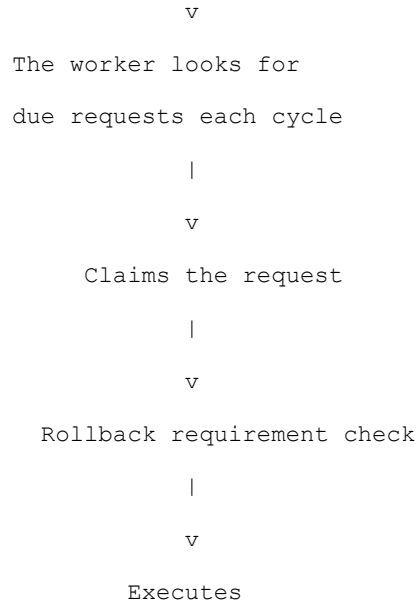
The worker also runs `ConnectionCheckJob` at regular intervals, trying to connect to every active server and updating the status. This makes an unreachable server visible before a request runs.

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 the request's 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.

Several servers can be selected only on a query result (QueryResult) request; change and rollback requests go to a single server, and the backend enforces this as well. The reason is that there is no distributed transaction across servers: if the first server committed and the second failed, a partial change would remain. Server names are stored in a single field, separated by semicolons:

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 is not counted against the license (Section 3.4).
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.

6.8 Session name on the target and the session probe

The product gives a name to every connection it opens to a target database. This lets the database administrator and the database activity monitoring (DAM) tool see whether a session belongs to the product and what job it belongs to. The name is given to the connection; the text of the script is not touched.

Name format: SQL Change Guard/<Role>/<Number>. Example: SQL Change Guard/Execute/123. The name is at most 63 characters and contains only ASCII letters, digits, space, slash, hyphen and underscore. In Oracle the name is assigned after the connection is opened: the role goes to MODULE, the number to ACTION and the full name to CLIENT_IDENTIFIER.

Role	When	Pool	Window
Execute	While a change request runs	Closed	Opens
Query	Query result request, per server	Closed	Opens
Release	Release package, AllOrNothing mode (single connection)	Closed	Opens
Sandbox, SandboxRelease	Isolated trial of a request or a package	Closed	Opens
Health	Connection health check by the Worker	Open	Does not open
Preview, Definition, Test	Column preview, object definition read, connection test	Open	Does not open

For the five roles that carry an identifier the connection pool is closed: the session closes when the work ends. For Health, Preview, Definition and Test the pool is open; a connection may stay idle for a while. This is normal and these connections do not run scripts.

After the connection is opened the product sends one read query to the target and learns the session number and the client address as the target sees it (SQL Server: session number and client address; PostgreSQL: process number and client address; Oracle: SID, audit session number and client address). The server says these two values, not the client, so they cannot be faked. If the query fails the fields stay empty, the work continues and a warning is written to the Worker log.

The application name is a label: the connecting side says it itself. The name alone is not treated as proof; the account, the address seen by the target, the session number and the time of the window are looked for together.

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: on which requests the change ticket number is mandatory. The change ticket is the change record the work is based on and comes from an external system (ITSM).
- Value: Always (required on every request) / ExceptEmergency (required on non emergency requests, not asked on emergency ones) / Never (not required)
- Default: Always
- Example: with Always, a request cannot be saved without a ticket number. ExceptEmergency exists because two flows live side by side in an organisation: a planned change starts in the ITSM system and arrives with its ticket, while the emergency intervention at midnight starts in the product itself and there is no change record yet at that moment. Without this option the rule would either be relaxed everywhere or the person on call would type a made up number into the field; the latter is worse than leaving it empty, because a number that cannot be verified in an audit creates the impression that a record exists.
- 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}$`
- Example: with the default, CR-12345 is accepted while 12345 and CR-123 are rejected.
- If empty: no format check is applied and any text is accepted.
- For your own organisation: if your change record numbers look like CHG0001234, use `^CHG\d{7}$`.

EmergencyTicketRequired

- What it does: whether the incident number is mandatory on emergency requests.
- Value: Always (required), Never (not required)
- Default: Never
- What the incident number is: the number of the incident record that caused the urgency. It is a different record from the change ticket. One describes the failure (the payment service stopped), the other the intervention (a corrupt index will be rebuilt). An auditor asks two separate questions: why was this change made at midnight with a shortened approval, and which record does this change rest on. The incident record answers the first.
- Example: with Always, an emergency request cannot be saved without an incident number. The field is not asked at all on requests that are not emergencies.
- Why the default is Never: the emergency path must never be blocked by a data entry rule the organisation has not deliberately adopted. It does not stay quietly off either: until a decision is made it appears as "Decide" in the readiness list on the Diagnostics screen.
- Control setting: yes. Relaxing it leaves the audit question "which failure does this emergency change rest on" unanswered.

EmergencyTicketRegex

- What it does: format validation of the incident number.
- Value: a regular expression. Can be left empty.
- Default: `^INC\d{7}$` (ServiceNow incident format, example INC0012345)
- For your own organisation: incident number formats differ between organisations (INC0012345 in ServiceNow, OPS-123 in Jira). The installation value is the ServiceNow format. If you use the same format, save the value once; otherwise write your own pattern or leave it

empty. Leaving it empty is a valid decision: the incident number is then accepted without a format check.

- Example: with the default value INC0012345 is accepted, while 12345 and INC-12345 are rejected.
- Independent of the change ticket: the two are separate records and their formats usually differ. The incident number does not have to match the change ticket pattern.
- Note: a pattern that cannot be compiled is rejected on save. A broken pattern written directly into the database does not lock the user out; the format check is skipped for that request and the situation is written to the log, and the readiness list shows it as "Attention".

Tickets on rollback requests

A rollback request inherits its traceability from its source: which request it rolls back is recorded, and its ticket comes from that source. A rollback request linked to a source is therefore not asked for a ticket number of its own. Otherwise, after an emergency change that could be opened without a ticket was executed, automatic rollback generation would hit the rule and the safety net would fail to be created on the riskiest change of all. A request that is not linked to a source does not benefit from this exemption.

RequestTicketRegex

- What it does: format validation of the number when a data extraction request is based on a service request (ItsmQueryResultTicketKind = Request). It also applies while ITSM is off.
- Value: a regular expression. When left empty, the number of a data extraction request is checked against ChangeTicketRegex; leave it empty if you do not use a separate service request format. Installation value: ^RITM\d{7}\$ (ServiceNow requested item format, example RITM0010001).
- Default: ^RITM\d{7}\$
- Example: with ^RITM\d{7}\$, RITM0010001 is accepted and CHG0030001 is rejected.
- Control setting: yes.

ITSM integration parameters

The parameters below set how the ITSM integration behaves. All of them are control settings. How they work is explained with examples in Section 22. The connection details (which ITSM, address, account) are not on this screen; they are in the Itsm section of appsettings.json.

Key	Values	Default	What it does
ItsmEnabled	true / false	false	Master switch. While off, the product does not contact ITSM.
ItsmVerifyOnSave	Block / Warn / Off	Block	What happens on save when a rule is violated (closed, missing or wrong type record).
ItsmExecuteGate	Block / Off	Block	Whether the change record must be ready right before execution.
ItsmEnforceWindow	true / false	false	Whether execution must fall inside the planned window in ITSM.
ItsmVerifyIncident	Block / Warn / Off	Warn	How the incident number on an emergency request is verified in ITSM.
ItsmQueryResultTicketKind	Request / Change / None	Request	Which record a data extraction request is based on.
ItsmUnavailablePolicy	AllowAndRecord / Block	AllowAndRecord	Whether work continues when ITSM cannot be reached. Emergency requests always pass.
ItsmReadyStates	State names (separated by ;)	Scheduled;Implement	States counted as ready.
ItsmClosedStates	State names (separated by ;)	Review;Closed;Cancelled;Cancelled;Resolved	States counted as closed.

Key	Values	Default	What it does
		d;Closed Complete;Closed Incomplete;Closed Skipped	
ItsmApprovedValues	Values (separated by ;)	Approved	Values of the ITSM approval field counted as approved.
ItsmWriteBackEnabled	true / false	true	Whether notes are written to the ITSM record.
ItsmWriteBackEvents	Event names (separated by ;)	Request events	Which request events are written as notes.

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.

QrDeliveryMode

- What it does: the channel through which the result file and the ZIP password that opens it are delivered. This value is the organisation CEILING: a request may pick a stricter mode, never a looser one.
- Value: PortalOnly, AttachmentPortalPassword or AttachmentAndPassword. Selected from a list on the screen.
- Default: AttachmentPortalPassword
- PortalOnly: neither the file nor the password goes by e-mail; only a "your result is ready" notification is sent. Both the file and the password are collected from the portal. This is the strictest option.
- AttachmentPortalPassword: the ZIP is attached to the e-mail, the password is NOT written in it and is collected from the portal. This is the default.
- AttachmentAndPassword: the ZIP and its password travel in the same e-mail. This is the loosest option and requires a written reason.
- Why the default keeps the password out of the mail: the situation an encrypted ZIP protects against (a mailbox being compromised, a message going to the wrong person, an archive being read) is exactly the situation where the password sits in the same message. With the password next to it, the encryption protects nothing.
- When a reason is required: when the mode that will be applied is AttachmentAndPassword. If the request picks no mode, the organisation ceiling applies, so a reason is still required.
- Viewing the password in the portal: a written reason is mandatory and a request without one is rejected. Every view is written to the audit trail.

- How the applied mode is decided: the stricter of the request's choice and the organisation ceiling wins. If the ceiling is tightened after the request was saved, the older request follows the new rule as well.
- Frozen at delivery time: the applied mode is written into the delivery record and is never recalculated afterwards. A later change to the organisation setting therefore cannot alter the report or the evidence dossier of a past delivery.
- Control setting: yes. Loosening it goes through dual control approval and is written to the audit trail as a separate event.

QrMaxRowCount

- What it does: the largest number of rows a query result request may deliver.
- Value: row count.
- Default: 1000000
- Where it applies: while the result is being read. The read stops the moment the limit is reached.
- Example: with a limit of 1000000, a request returning 1,200,000 rows cannot run. The request fails, the reason is written to the execution log, no package is produced and no e-mail is sent.
- Why it applies during the read: reading everything first and rejecting afterwards saves neither time nor memory. In measurements a two million row result filled 1.1 GB of memory before it would have been rejected. Because the limit applies during the read, more rows than the limit are never held in memory, and masking and file writing never run at all.
- Relationship with Excel: when the result is written as Excel, the effective limit is the smaller of this value and the Excel format limit of 1,048,575 data rows. That second limit is not a preference; it is a rule of the xlsx file format and cannot be changed.
- Control setting: yes. Raising it means a single approved request can carry more live data out of the organisation.

QrOutputFormat

- What it does: the format of the data file in the result package.
- Value: Auto, ExcelOnly or CsvOnly. Selected from a list on the screen.
- Default: Auto
- Auto: Excel when the row count is below QrCsvThresholdRowCount, CSV when it is above.

- ExcelOnly: always Excel. The effective row limit intersects with the Excel format ceiling of 1,048,575 rows.
- CsvOnly: always CSV. The Excel row ceiling does not apply; only QrMaxRowCount does.
- CSV format: the separator is the semicolon and the file is written with a UTF-8 marker, so it opens in the correct columns and without broken characters.
- Security note: TEXT cells starting with an equals sign, plus, minus or @ are neutralised so they do not run as formulas when the file is opened in Excel. The value itself is not lost, only its execution is prevented. Numbers, dates and boolean values are not affected, so negative numbers stay numbers and can still be summed.
- Control setting: no. The format changes neither who can reach the data nor how much data leaves. In both formats the package is delivered inside the same encrypted ZIP with the same masking applied.

QrCsvThresholdRowCount

- What it does: the switch point between Excel and CSV in Auto format.
- Value: row count.
- Default: 500000
- Example: with 500000, a 400,000 row result is written as Excel and a 600,000 row result as CSV. At exactly 500,000 rows Excel is written, because the threshold means "more than this".
- First limit: the value cannot be greater than QrMaxRowCount. If it were, it could never take effect: the result is already cut at the upper limit, so the threshold would never be passed and the setting would be silently meaningless.
- Second limit: the value cannot exceed the Excel ceiling of 1,048,575 rows; if a higher value is entered, the ceiling is applied instead. Otherwise results between the ceiling and the threshold would not go to CSV but would be written to Excel, and would fail at the very end of the file write.
- Control setting: no.

QrExcelAutoFitMaxRows

- What it does: the largest row count for which column widths in the Excel file are fitted to their content.
- Value: row count.

- Default: 50000
- Example: with 50000, columns in a 20,000 row file are widened to fit their content; in a 200,000 row file this step is skipped and columns keep their default width.
- Why there is a threshold: fitting measures font metrics and in measurements it consumed about a third of the Excel write time on its own. It affects appearance only, never the data itself.
- Control setting: no.

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.

7.11 SIEM group

This group decides how much of the governance activity reaches the SOC collector. The connection details are not here but in the appsettings file (Section 2.4.10); this screen manages only what gets sent.

All four are control settings. The reason is plain: changing them is the only way an organisation could hide itself from the SOC. Closing a stream, raising the threshold or turning off the master switch silences the trail that is visible from outside. The script text flag is a control in the opposite direction; turning it on can push business data to the SOC. Both belong on the same list, because both decide what leaves the building.

Parameter	What it does and what to enter	Default
SiemEnabled	Master switch. While it is off no event is sent. It is on (true) after installation. Delivery needs at least one endpoint defined in appsettings (Section 2.4.10); if none is defined the Governance Setup Check screen shows it as missing.	true
SiemStreams	Streams to send, separated by semicolons. There are four: Audit is the audit trail, Anchor carries the ledger digest events (digest published, digest failed, restore, fork), AppLog is the product's own log and Heartbeat is the heartbeat. Leave it empty to send all four. An example of narrowing: Audit;Anchor;Heartbeat	Audit;Anchor;AppLog;Heartbeat
SiemMinSeverity	Events below this severity are not sent. Enter one of Info,	Notice

Parameter	What it does and what to enter	Default
	Notice, Warning, Error and Alert. Narrowing the scope is legitimate because SIEM licences are priced by volume; however five classes of event are not affected by this threshold and are always sent (see below).	
SiemIncludeScriptText	Whether the full script text goes to the SOC. The default is off, because a script may contain business data. While it is off only the script digest is sent, which is enough to show that the script is the same text that was approved. Check the SOC data classification rules before turning it on.	false

Events that cannot be silenced by the threshold

Whatever SiemMinSeverity is set to, the five classes below are sent. The first four are the SOC side counterpart of the never skip rule in the risk model.

Emergency actions. Emergency override, step skipping and similar moves. They do not remove approval, they shorten it; the trail must remain.

Permission denials and failed authentication. A refused action matters as much as an allowed one; the first sign of an attack usually shows up in refused requests.

Control weakening events. Changes to approval policies, SQL standards, the critical object list, parameters, roles, servers, licence and settings approval, together with creating users, deleting them, changing roles and revoking sessions. These are the most valuable events for the SOC, because from inside the product they look ordinary.

Ledger digest events. Digest published, digest failed, restore and fork. They are the external witness of the evidence and the volume is a few messages an hour; even at a low severity they cannot be filtered out. Restore and fork are also raised at alert level.

The heartbeat. Its severity is deliberately low, because it is not an event but a sign of life and should not appear among events on the SOC dashboard. If the threshold could silence it, it would be useless: the default threshold already sits above the heartbeat level, so the heartbeat would never be sent on any installation. The legitimate way to turn the heartbeat off is to remove the Heartbeat stream from the SiemStreams parameter; that path is written to the audit trail and announced to the SOC, so turning it off stays visible.

Narrowing the scope is announced to the SOC

When one of these four parameters changes, a separate event is sent to the SOC using the configuration that was in force before the change took effect. The order has to be this way: if the announcement used the new configuration, a change that closes the stream could also silence its own announcement.

The four eyes rule does not break this flow. A change that needs approval is not applied on the first save; it is applied the moment it is approved, and the announcement goes out at that moment.

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, QrDeliveryMode, QrRequesterCanDownload, QrDownloadMinimumRole, QrFileRetentionDays, QrMaxRowCount, QrEmailApprovalPolicy, QrEmailApproverRole, QrEmailAutoApproveUsers, RequesterCanSelfApproveExecute, ApproverCanExecute, RequireDistinctApproverPerStep, EmergencyMinimumRole, RejectOnError, ChangeTicketRequired, EmergencyTicketRequired, ChangeTicketRegex, EmergencyTicketRegex, RequestTicketRegex, all ITSM integration parameters (ItsmEnabled, ItsmVerifyOnSave, ItsmExecuteGate, ItsmEnforceWindow, ItsmVerifyIncident, ItsmQueryResultTicketKind, ItsmUnavailablePolicy, ItsmReadyStates, ItsmClosedStates, ItsmApprovedValues, ItsmWriteBackEnabled, ItsmWriteBackEvents), the organisation declarations (WorkerEndpointNetworkRestricted, TargetAccountDedicated, DamAllowlistMaintained) and the SIEM group (SiemEnabled, SiemStreams, SiemMinSeverity, SiemIncludeScriptText).

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 Settings Approvals screen

Screen: Settings > Settings Approvals

Column	What it shows
Screen	Which screen the change belongs to (server, parameter, policy and so on).
Operation	Insert, update or delete.
Record	A readable summary of the change; the record key when there is no summary.
Requested by	The user asking for the change.
Requested at	Time of the request.
Status	Pending, Approved, Rejected, Failed.
Actions	The Detail button at the end of the row. Opens the current value next to the requested value.
Decided by (not a separate column)	The user who approved or rejected. Shown in small text under the Status column together with the decision time.

Column	What it shows
Reason (not a separate column)	Rejection reason or failure cause. Shown in the detail dialog.

The Detail button at the end of the row shows the current value next to the requested value. The buttons above the list switch between Pending, Approved, Rejected and Failed records.

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. Can be left empty; when empty it covers every database.	No	CustomerDB
Object Type	Table, procedure, view, function, trigger.	Yes	Table
Schema Name	The schema of the object. Can be left empty; when empty it covers every schema.	No	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 88 standards. Each can be enabled or disabled and its severity changed. Naming rules are held in separate rows per statement form: there are separate keys for CREATE, ALTER and CREATE OR REPLACE. They are parts of a single rule. When you update the base key, its active state, prefix and severity are copied to the other forms as well, and those rows state which rule they follow. Without this link the rule would be broken silently: on Oracle and PostgreSQL servers objects are almost always written with CREATE OR REPLACE, so if only the CREATE form were enabled the rule would never apply on those servers.

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.	

Field	What it does	Values
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

Key	What it checks	Severity	Never skip
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.
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.	

Field	What it does	Values
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.

Column	What it shows
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 the organisation parameter is used.
Distinct approver per step	Same.	Empty means the organisation parameter is used.

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

A request cannot be created for a server that has no policy. The save is rejected and the error message names the server that is out of scope. The reason: if no policy matches, no approval step is produced, nobody can approve a request without steps, and the request would enter a state with no way out.

Rollback requests are outside this rule. A rollback is a recovery operation; it is saved even when it falls outside policy scope, because blocking it here would also prevent the automatic rollback from being produced when the source change is approved, leaving nothing to roll back to during an incident. When no policy matches, the product mirrors the approval requirement of the source change: the role authorised to approve the change also approves the rollback.

Installation recommendation: define at least one global policy. Otherwise no request can be created for servers left out of scope.

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's role meets the step's approver role; otherwise seniority would remove the control from the person doing the risky work. Rule 4 (SkipIfRequesterRoleAtLeast) is set explicitly on the step, so it also applies to a triggered step; locked steps are not affected by it either. Rules 4 and 5 depend on the person and do not run at all while RequesterCanSelfApproveExecute is off; otherwise a request from someone who may not approve their own request could pass with no approval step left.

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	Queued for execution or running in the Worker.	No
8	ReconciliationRequired	Waiting for reconciliation: the execution reached the target but its outcome is not known for certain. It is not executed again; an authorized person checks the target and records the outcome.	No

What a terminal state means: the request has left the active flow. The worker does not pick it up; the request cannot be edited, approved, executed or scheduled. The few actions still possible (such as copying) are listed in the table in Section 11.5. Only the owner can edit a request in Pending or Approved; when an approved request is saved again its approvals are reset.

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

Current state	Possible action	New state
Approved	Execute	Executing; then Executed. On error Rejected (installation value) or Approved depending on RejectOnError; ReconciliationRequired if the outcome is uncertain
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
Executed	Mark done	Executed (information updated)
Rejected, Manual, NoLongerNeeded	None	-
Executing	(recovery) Attempt with an uncertain outcome	ReconciliationRequired
ReconciliationRequired	Record the outcome: the change is on the target	Executed
ReconciliationRequired	Record the outcome: the change is not on the target	Approved

When execution fails and everything on the target is rolled back, the behaviour depends on RejectOnError: true gives Rejected, false returns the request to Approved. In every case where the target may have committed (the running Worker was interrupted, the result could not be recorded, the script ended the product's transaction itself) the request is not rejected and does not return to Approved; it moves to Reconciliation Required (ReconciliationRequired). Such a request cannot be edited, executed, scheduled or added to a package. A person with execute permission checks the target and, with the Record the outcome button in the request detail, states "the change is on the target" (the request becomes Executed and the script does not run again) or "the change is not on the target" (the request becomes Approved), and writes in the note what was checked. The decision, the note and the session of the decision are written to the audit trail with the ChangeRequest.ReconciliationResolved event. The "Executions with an uncertain outcome" row on the dashboard counts these requests.

Partial outcome: if only part of the change is on the target (for example the first block became permanent and the following blocks did not run), "the change is on the target, in full or in part"

is chosen, because running the request again would write the part that reached the target a second time. The note says what was applied and what was not. The remaining work is sent in a new request opened with the "New request for the remaining work (copy)" button in the request detail. The new request goes through approval, risk and audit again and is recorded as linked to the first one: the detail of the new request shows "Copied from request" and the detail of the first request shows "Requests copied from this request".

After the decision, the Reconciliation Record box in the request detail shows why the request was moved to reconciliation, the decision, who made it, when, and what was checked on the target. This information is read from the evidence ledger; the Reconciliation section of the evidence file carries the same information.

Additional check on Oracle targets: on Oracle every DDL statement (such as CREATE, ALTER, DROP, TRUNCATE, GRANT), a COMMIT in the script and a COMMIT inside a procedure make the data changes made so far (INSERT, UPDATE, DELETE, MERGE) permanent. If an error follows, rolling back in SQL Change Guard cannot undo them. For this reason, on Oracle the product reads the transaction id before and after every block and tracks whether a data change has become permanent.

If an error comes after a data change has become permanent, or a block ends the transaction itself while data changes are pending (for example a procedure that contains a COMMIT), the execution stops and the request moves to Reconciliation Required. The request is not executed again; running it again would write the same data a second time. If the script rolls back pending data changes with ROLLBACK, the execution stops and the request stays a normal failed execution. If a script that contains only DDL fails, the earlier DDL statements are permanent by Oracle's own behaviour; the execution log says so and the rollback request is used if needed.

In a release package the same rule applies to the shared transaction of the package: if a data change of an earlier request has become permanent and a later request fails, the requests of the package that ran up to that point and the failing request are moved to reconciliation. At save time the OracleDmlBeforeImplicitCommit rule blocks DDL that follows a data change in the same script; the check at run time catches what cannot be seen in the script text (a COMMIT inside a procedure, dynamic SQL, the effect of one request on another in a package). Known limit: the COMMIT of a procedure marked PRAGMA AUTONOMOUS_TRANSACTION does not touch the product's transaction and cannot be tracked.

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 server. A change or rollback request targets a single server; several servers can be selected only on a query result request, because there is no distributed transaction across servers.	Yes	See Section 6.6.
Ticket Number	ITSM ticket number.	Depends on ChangeTicketRequired	Format is validated with ChangeTicketRegex; on a data extraction request RequestTicketRegex applies when it is set. While ITSM is on, the number is also verified in ITSM (Section 22).
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.

Prerequisite: the server must have an approval policy

Before a request can be created, the selected server must be covered by an approval policy for that request type. If a server outside any policy is selected the request is not saved and you get this error: no approval policy is defined for this server, a policy must first be matched to it on the Settings > Approval Policies screen.

The reason for the rule is this: when no policy matches, the request produces no approval steps at all. Nobody can approve a request that has no steps, because the problem is not the user's role but the absence of anything to approve. Not even the most privileged user can approve it. Such a request would enter a state with no way out the moment it was created; the rule cuts that off at the source.

Several servers can be selected only on a query result request. On such a request all of them must be covered. One covered server does not rescue the others: otherwise the uncovered server would run under a policy that was never written for it, and no approval decision would have been made for that server at all. The error message names the servers that are missing, so you do not have to look for them.

This is a one time setup task. Once a policy is bound to a server, or to the environment that server belongs to, or globally, that path is settled for good. To see at a glance which servers are outside every policy, look at the Active servers with no approval policy row on the Settings > Diagnostics screen; while that row reads zero, no server can hit this rule.

Two cases are outside this rule. Emergency requests go through a separate check: an emergency policy must be bound to the selected server before a request can be declared an emergency. Rollback requests are never blocked. A rollback is a recovery action; if it were blocked for having no policy, there would be nothing left to roll back to during an incident. Automatic rollback generation goes through the same save path, so without this exemption the rollback script could not be produced and the failure would be silent.

Removing a policy binding does not affect requests that already exist, but no new request can be created for that server afterwards. It is worth checking whether any request is still pending on that server before removing its binding.

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:

- A syntax error always stops the save. For other error level findings the request is saved if the user confirms on the second attempt, and the `ChangeRequest.ParseErrorsConfirmed` event is written. Save-blocking standards cannot be passed even with this confirmation.
- 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.

Affected objects and the difference view

When parsing finishes, the lower panel shows two tabs: Parse Results and Affected objects. The second tab lists every object the script touches, row by row: object name, statement type, schema, database and the script file it came from.

The View diff button at the end of each row opens the CURRENT version of the object on the target server side by side with the version in the script. It answers a single question: what is on the server before this request is applied.

In the window, the left side is the definition on the server and the right side is its counterpart from the script. You can switch servers from the toolbar; the list only offers active servers of the same database type as the request, because the same object cannot be looked up on a server of a different type.

What appears on the left depends on the object type:

- Procedure, function, view, trigger: the source text stored on the server.
- Table: a CREATE TABLE script produced from the columns, constraints and indexes on the server. A table has no stored source text, so the definition is generated.

What appears on the right depends on what the script does. With CREATE or ALTER PROCEDURE the script is the new full version of the object, so a line by line comparison is directly meaningful. With statements such as ALTER TABLE ADD COLUMN the script holds only the change to be made; in that case the heading on the right says it is a statement and no line by line difference is expected. The left side is still useful: it shows whether the column already exists, its type and the constraints.

If the object does not exist on the server, that is not an error: it may be a new object. The left side stays empty and the window shows a Not on server note.

If the script does not state a database name (no USE statement on SQL Server), the product searches the databases on the server. So that you can see which database was used, the database is written in the heading of the left column. Because an object with the same name can exist in several places, writing the USE statement in the script is recommended whenever the target database matters.

The same ambiguity applies when the schema is not written. The product first tries the schema of the connected user (the first entry of the search path on PostgreSQL, the session schema on Oracle); writing the schema in the script is still the safest choice.

The only case where no comparison is possible is when the parser could not extract the script block for that row. The button is disabled there and the tooltip states the reason.

How to check: write a script that changes an existing procedure (ALTER PROCEDURE), run the parse and press View diff on the Affected objects tab. The current body on the server appears on the left, the body you wrote appears on the right and the changed lines are marked. Repeat with a script that adds a column to an existing table: the left side shows the current CREATE TABLE equivalent of the table.

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.

Field	What it shows
Status	One of the states in Section 11.1.
Request type	Change, QueryResult or Rollback.
Risk band	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 operation on the object is destructive.	The operation counts as risky when it is DROP or TRUNCATE. It is independent of standard findings and their severity.	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 is destructive by nature (DROP, TRUNCATE).

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, a complete rollback exists if the server requires one, a current successful sandbox exists if the server requires one, segregation of duties rules do not block
Schedule	Approve permission and the scheduling module	Status Approved; the same rollback, sandbox and segregation of duties conditions as Execute
Cancel schedule	Approve permission	Status Approved and a schedule exists

Button	Visible when	Enabled when
Mark done	Approve permission	Status Executed
Mark manual	Approve permission	Status Pending or Approved
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 largely prevents the "the button was enabled but the action failed" situation. Checks made at the moment of the action, such as the approval chain, ITSM and the evidence write boundary, can still stop the action while the button is enabled; the reason is given in the message.

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

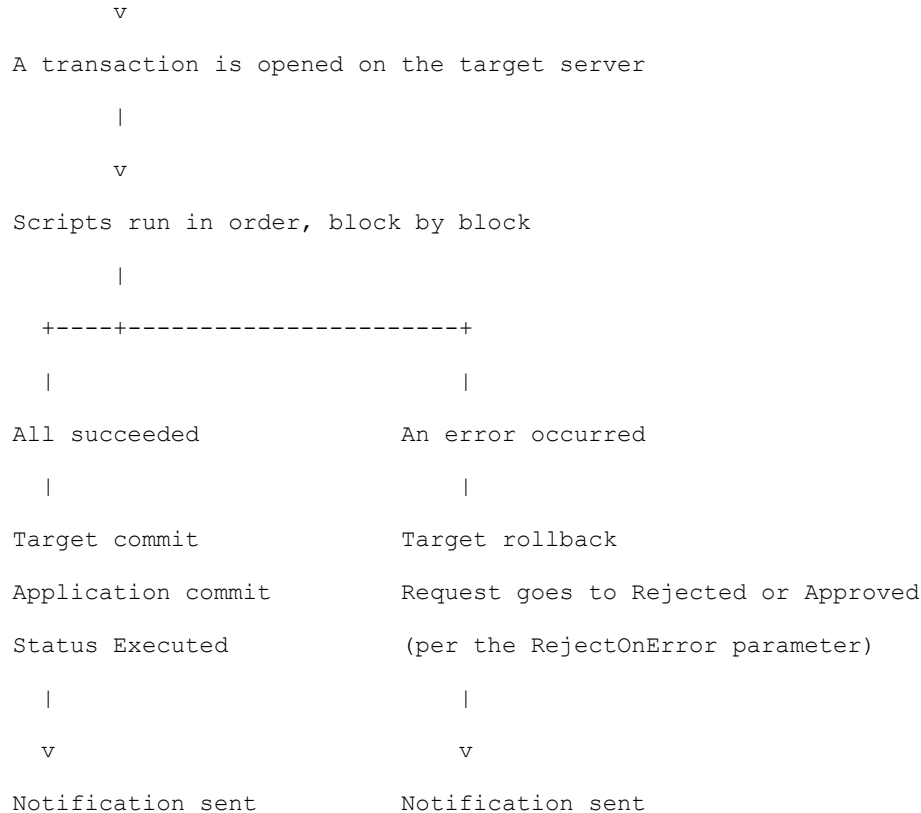
ExecutionRequestedBy = the person who pressed the button

|

v

The worker claims the queued request

|



Scheduled execution

In manual scheduling the user picks a date. It must be at least `ExecutionDelaySeconds` in the future (installation value 300 seconds; at least 30 seconds applies even if the parameter is set lower). 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`. Who can request a cancellation: any user with the `execute` permission, being the person who started the execution is not required. The reason is that a cancellation is not an ownership action, it is a brake. Stopping an execution that is going wrong in production cannot depend on the person who pressed the button being present: that person may be in a meeting, their session may have dropped, or their permission may have been revoked in the meantime. The restriction on executing your own request does not apply to cancellation either; even when a requester cannot execute their own request, they can stop a running operation if they hold the `execute` permission. Accountability is preserved, because the person who started the execution and the person who requested the cancellation are kept separately and both appear in the audit trail.

Crash recovery

If the worker service stops mid-run, the request stays in Executing. Such requests are detected against a time threshold. A request that certainly did not commit on the target returns to Approved; a request that may have committed on the target is moved to Reconciliation Required (ReconciliationRequired) and is not run again (Section 11.1). The threshold is the Workers:StaleExecutionRecoveryMinutes setting and defaults to 120 minutes. For details and how to choose the value see Section 25.7.

Approval chain: what was saved, approved and executed is the same

Every time a request is saved, everything that determines its content is gathered in one fingerprint: the order and text of the scripts, the address, database and connection account of the target servers, the request type and, in a query result request, the recipients, the delivery mode and the unmasked columns. The description, ticket and schedule are not part of the fingerprint. The script texts are also written to dbo.ScriptContents; this is an append-only ledger table and the database engine rejects UPDATE and DELETE on it.

The save, the approval plan, every approval step, the final approval and the execution events carry this fingerprint. Every event is linked by its number to the event it rests on; when evidence signing is on (Evidence:Signing:Enabled, off in the code default) it is signed with the product's evidence key. The approval plan carries the steps at approval time and a digest of the organisation rules. The chain shows that the application accepted and recorded the event; on its own it does not prove the approver's intent or what actually happened on the target.

The Worker checks the chain before it connects to the target. If the script, the target server, the recipients or the approval steps were changed outside the application (directly in the table) after approval, the fingerprint does not match. If the approval plan or a signed approval step is missing, or a signature is invalid, the chain is broken. In these cases the target is never contacted, the attempt is recorded with its reason as a ChangeRequest.ExecutionBlocked event and the request returns to Approved without being rejected. If the request is saved again in the application, the approvals are reset and the new content is approved again.

The same check runs before approval: if the content the approver saw does not match the last save, the approval is not given and a ChangeRequest.ApprovalBlocked event is written.

Approval policies, the parameters that affect approval, the never-skip standards and the user roles are sealed after every authorised change. If these tables are changed outside the application, no new approval plan is built and a user whose role looks different cannot approve; the attempt is recorded with its reason as a ChangeRequest.ApprovalBlocked event and goes to the SOC when SIEM is on. A change made on the settings screens in the meantime does not accept the state of the tables; it writes a Governance.StateMismatch alert. After reviewing the difference, a SystemManager rebaselines with a reason on Settings > Diagnostics > Governance

Readiness. This is recorded as a Governance.Rebaselined event and goes to the SOC as an alert when SIEM is on. When a request whose plan could not be built is saved again, a new plan is built.

The content of a release package (requests, order, dependencies, execution mode, server) is also sealed when it is saved and queued. A package changed outside the application is not executed and a ReleasePackage.ExecutionBlocked event is written.

Execution with an unsigned chain on a production server is off by default (Evidence:Signing:AllowUnsignedProductionExecution = false). The chain status is shown by the Verify Script Integrity button on the request detail and in section 5b of the evidence dossier. While signing is off, execution on a Production target is refused (SIGNING_DISABLED_PRODUCTION); evidence signing must therefore be turned on for production use. On non-production targets an unsigned chain is accepted.

11.7 Partial success and rollback

A request contains several blocks. If an error occurs:

Database	Behaviour
SQL Server, PostgreSQL	The product's transaction is rolled back; if the script did not end the transaction itself, 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, on Oracle targets an active and complete rollback request is mandatory for requests containing DDL, DCL or TRUNCATE; even when the server does not require rollback, such a request does not run without one.

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 generated and executed only after the source request has been applied. The exception applies to Oracle targets only: because Oracle commits DDL implicitly, blocks that ran before the error may remain in production, so a rollback can be generated and executed even for a closed request whose execution failed. There is no such exception on SQL Server and PostgreSQL targets: execution runs inside a single transaction, every block that ran is rolled back on error, and no change is left to roll back. On those targets the button is disabled and the tooltip states the reason. The rule was narrowed because otherwise the generated rollback script could drop a change made by a different request that created the same object.
- Rollback requests are never scheduled automatically. A human must always execute them.

How the rollback script is produced

The product produces the rollback script by looking at what the change does. The rollback of a change that adds a column drops that column, and the rollback of a change that drops a column adds it back. Most of these operations can be derived from the script itself. No script is produced for data changing statements (INSERT, UPDATE, DELETE, TRUNCATE, MERGE) or EXEC: the product writes a manual intervention note marked TODO at that point and the rollback counts as incomplete. If data must be rolled back, the requester writes that script. This behaviour is the same on all three target engines.

A change that drops a table (DROP TABLE) is different. A database has no command that brings back a dropped table. The product therefore connects to the target server before the table is dropped, reads the definition of the table and produces a CREATE TABLE script from it. The rollback here is not a rewind; it is a rebuild.

This distinction matters, because a rebuild covers only what can be read from the server.

What the rebuild covers

Restored: columns and data types, the automatic increment (IDENTITY) setting, nullability, default value constraints, the primary key, unique constraints, check constraints, foreign keys (including their delete and update behaviour) and indexes.

Not restored: the formula of a computed column (on SQL Server and PostgreSQL targets; Oracle keeps the formula). On SQL Server targets the triggers on the table, extended properties (description fields), the partition scheme and the filegroup placement are also not restored.

A computed column is a special case and should not be approved without knowing the outcome. On SQL Server and PostgreSQL targets the column comes back, but it comes back as a plain column: it has no formula, it stops calculating itself and it stays empty. The guard that prevented writing data into that column is also gone, so an insert that used to fail no longer fails and any value can be written into the column.

This list is not a defect; it is the limit of the method. The script runs and the table comes back. The product does not hide the limit: it puts a note at the end of the generated script and shows the same information on the request screen.

The two different warnings on screen

There are two separate warnings about rollback scripts. They say different things and must not be confused.

The first one is a blue information banner and describes the limit of coverage. It appears at the top of the request detail screen. It means the rollback script is ready and will run, but it does not restore the objects listed above. This banner does not block execution; it exists so that the approver knows what is being approved. The banner appears when the script contains the coverage note, which in practice means the table is rebuilt, that is, the rollback of a change that drops a table. It does not appear for the rollback of a change that adds a column, because no table is rebuilt there.

The second one is the yellow Required steps that were not completed panel and describes a real gap. The line A rollback plan exists but its script is incomplete means that a manual intervention note marked TODO remains in the script. There are two reasons: the change contains a data changing statement (INSERT, UPDATE, DELETE, TRUNCATE, MERGE or EXEC; no script is produced for these) or the product could not connect to the target server and read the definition of the object. The server may be unreachable, the object may not exist on that server, or the user may not have enough read permission. In these cases the product cannot write a working command into the script; it leaves a comment line instead, and that comment is all there is.

The difference is this: with the blue banner the script runs and does part of the job, with the yellow panel the script does nothing.

Consequences of an incomplete rollback script

If a rollback is required in the server settings, a request whose rollback script is incomplete cannot be executed. The Execute button is disabled and the reason is stated on screen. If a rollback is not required, execution is not blocked, but the warning still appears. Not being required does not mean there is no problem.

The way to clear the warning is to complete the script. Edit the rollback request, write the real command where it is missing and save. The warning is not a stored state; it is computed on every read from the current content of the script, so it clears by itself as soon as the script is complete.

Note one point: the product checks whether the script carries an incomplete marker, not whether the script does the right job. Deleting the comment line and writing nothing in its place also clears the warning. Making sure the script really performs the rollback is the responsibility of the person reviewing it.

When a rollback script is changed, the approval of the request is reset and a confirmation is requested while saving. A changed rollback cannot carry the old approval.

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.

To declare an emergency the user's role must be at least the EmergencyMinimumRole level (installation value Requester).

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.
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, favourites, only my requests and emergency changes awaiting review. The last three are two state buttons: pressed means the filter is on, pressing again turns it off. The flame button shows emergency requests that were applied but not reviewed afterwards; the counter on the dashboard turns the same button on when clicked. Arriving from the dashboard does not apply the other filters, so the number on the dashboard and the number of rows in the list match.

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

Rows carry small badges in addition to the columns. The RB badge shows that the request has a rollback plan. A grey RB means the plan is ready. An orange RB means the plan exists but its script is incomplete; hovering over the badge states the reason. If there is no rollback plan at all, no badge appears.

The RB badge appears on change requests only. There is no rollback concept for query result requests.

Section 12. Risk Band

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 is computed and stored in the record, but it is not shown on any screen, report or evidence dossier: two numbers describing the same idea leave the reader wondering which one to trust.
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 (for the record)

The score is a number between 0 and 100 and a higher number means safer. This section explains how it is computed for the sake of record integrity; the score is not shown to users, it is only stored in the database. 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
DropColumn	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, DropColumn, 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. The scores in this table are stored values, not something the user sees.

12.6 The score explanation (ScoreExplain)

The band alone is not enough: the user must see WHY it sits at that level. The system produces an explanation for every request:

Field	What it shows
Band	The computed risk band.
Score	The 0-100 number. Stored on the record, not shown on screen (Section 12.1).
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.

Eight standards are flagged never-skip at installation: CriticalObjectDrop, CriticalObjectAccess, CriticalObjectSchemaChange, XpCmdShellUsage, DclChangeDetected, LinkedServerUsage, NoManualCommitTransaction and NoManualRollbackTransaction. The last two protect the atomicity promise of the product: every block of a request runs inside a single transaction and all of them are rolled back on error. A COMMIT of the script closes that transaction early and a ROLLBACK of the script aborts it; either can leave a half applied change behind. At installation these two also block the save; when they fire the request is not saved. Opening a nested transaction (NoManualBeginTransaction) is

not flagged never-skip, but at installation it also blocks the save. The full list is visible on the Settings > SQL Standards screen.

What happens if someone tries to skip it: there is no way inside the product to skip a locked step; skip rules, role seniority and emergency override cannot pass it.

12.8 Where the risk band appears

Place	How it appears
Request list	Band badge.
Request detail	Band, finding list, explanation.
Dashboard	Risk distribution chart.
High Risk Changes report	Band column.
Evidence dossier	Risk summary and finding list.

12.9 How the band 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.
- Columns detected as sensitive are masked under the default policy (AutoMask).
- 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. This prevents bypassing the catalog with an alias. For queries whose source columns cannot be derived (dynamic SQL, some procedures) protection relies on content scanning at execution time.

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".

Each column in the list carries an expected decision. There are five states. Will be masked: the column matched the classification catalog and is masked on delivery. Checked by content: not in the catalog, its content is scanned at execution time and it is masked if a sensitive pattern is found. Could not be determined: the columns the query returns could not be derived (dynamic SQL and some procedures). Unmasked requested, awaiting approval: the request asked for this column unmasked but the unmasked approval has not been given yet. Will not be masked: the column was requested unmasked and the approval was given, so it is delivered in the clear.

The difference between the last two matters: asking for a column unmasked does not open it on its own. If the request runs before the unmasked approval is given, the column is still masked. That is why the screen makes no promise until the approval exists; it only states that the request was made. If an asterisk was written instead of column names, every column falls under the same rule; rows whose columns could not be determined stay outside that scope even with an asterisk, because it is not known which columns will be returned.

These two states are not stored in the record, they are computed when the screen is served. The reason: the unmasked approval is given after the request is saved, so a value frozen at save time would keep showing the wrong thing once the approval arrived.

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 written to an Excel or CSV file (see 13.7)
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.

 The auditor sees the unmasked delivery in this record.

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. The summary record goes to the QrMaskingAuditLog table and the detail record to the QrMaskingAuditDetail table:

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 or CSV file inside an encrypted ZIP; the format is managed with QrOutputFormat (see 13.7).
Password	Generated separately for every delivery.
Where the password lives	Stored encrypted on the request record (AES-256-GCM). An authorised user views it after entering a reason; every view is recorded as a QrZipPasswordView event.
Email	Depends on the delivery mode: PortalOnly (both file and password are taken from the portal only), AttachmentPortalPassword (default; the file is attached to the email, the password is only in the portal), AttachmentAndPassword (file and password go in the same email, requires a reason). 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 requires a confirmation box and a reason. The download is written to the audit trail as QrDownload (QrDownloadByNonRequester when the user is not the requester) and appears in the Data Access and Export Log report. A user who is not the requester can download the file if their role is QrDownloadMinimumRole (installation value ChangeManager) or higher; they do not need to be on the recipient list. Set this threshold according to the organisation's data access policy.

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.

13.7 Result file format and row limit

The data file in the result package can be Excel (xlsx) or CSV. The choice is made with the `QrOutputFormat` parameter and the default is Auto: small results are written as Excel because it is easier to read, large results as CSV.

The reason for the split is that Excel has two limits. The first is a format limit: a worksheet holds at most 1,048,575 rows besides the header row. This is not a preference; it is a rule of the xlsx file format. The second is cost: writing Excel becomes expensive quickly as the row count grows, while CSV is written as a stream and stays far cheaper. The measured figures are in the next section.

The row limit is applied WHILE the result is read. The moment the limit is reached the read stops, the request fails and the reason is written to the execution log. No package is produced and no e-mail is sent. This way more rows than the limit are never held in memory, and masking and file writing never run at all.

The effective limit depends on the selected format:

- CsvOnly: only `QrMaxRowCount` applies; the Excel ceiling does not.
- ExcelOnly: the smaller of `QrMaxRowCount` and 1,048,575 applies.
- Auto: results above the threshold go to CSV, so `QrMaxRowCount` applies on its own. If no threshold is defined every result is written as Excel, so the Excel ceiling applies again.

The CSV file is written with a semicolon separator and a UTF-8 marker, so it opens into the correct columns and without broken characters. Dates and decimal values are written in a fixed format that does not depend on the server locale: the same request produces the same file on two different machines, which keeps it comparable in an audit.

Formula neutralisation is applied to text cells. A text starting with an equals sign, plus, minus or @ would run as a formula when the file is opened in Excel; those cells are marked so they are read as text instead. The value itself is not lost, only its execution is prevented. Numbers, dates and boolean values are not affected, so negative numbers stay numbers and can still be summed in the file.

13.8 Measured cost and performance

The figures below are real measurements, not estimates. Measurement machine: Intel Core i5-8265U at 1.60 GHz, 4 cores and 8 logical processors, 8 GB of memory, Windows 11, .NET 10. On a server class machine these times are noticeably shorter; the point here is not absolute speed but where the cost goes.

Scenario: a result set with 10 columns, 5 of which are detected as sensitive and masked (national identity number, phone, e-mail, card number and IBAN). The rule set contains 14 enabled data patterns and 103 active catalogue rules.

Two million rows, 10 columns, 5 masked columns:

- Sensitive data detection: 0.4 seconds
- Masking (5 columns, 10 million cells): 28.7 seconds
- Writing the CSV file: 13.7 seconds, file size 267 MB
- Writing the Excel file: NOT POSSIBLE. The Excel format limit of 1,048,575 rows is exceeded, so the file cannot be produced.
- Result data held in memory: 1,089 MB; peak process memory 1,558 MB
- Packaging total (detection, masking and CSV writing): about 43 seconds. This does not include the time the query spends running on the server and the rows travelling over the network; that time depends on the query and the server.

The same scenario at smaller sizes:

- 200,000 rows: detection 0.01 s, masking 2.0 s, CSV write 0.8 s (26 MB), Excel write 28.2 s (7.6 MB), memory 500 MB
- 1,000,000 rows: detection 0.04 s, masking 10.8 s, CSV write 6.4 s (132 MB), Excel write 146.8 s (38 MB), memory 1,558 MB
- 2,000,000 rows: detection 0.4 s, masking 28.7 s, CSV write 13.7 s (267 MB), Excel write not possible, memory 1,558 MB

There are three conclusions to draw from this.

First, sensitive data detection is not expensive. Detection scans a sample rather than every row (1000 rows by default), so its duration does not grow with the row count. Even at two million rows it is about one percent of the total cost. There is no performance reason to avoid tightening the detection settings.

Second, the real cost is masking and file writing, and both grow in direct proportion to the row count. Masking costs roughly 2 microseconds per cell, so the time grows proportionally with the number of masked columns.

Third, Excel is not the right tool for large results. At one million rows an Excel write takes about 23 times as long as CSV (146.8 seconds against 6.4 seconds), and above 1,048,575 rows it does

not work at all. This is exactly what the Auto format is for: it looks at the size of the result and picks the right file format itself.

Sizing guidance: packaging a one million row result raises process memory to about 1.5 GB. The machine running the worker needs that much free memory for each query result request that may run at the same time. This calculation should be done before raising QrMaxRowCount.

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 can be seen here. A sandbox is not a full simulation of production conditions (data volume, concurrent load, locking). In the code default, sandbox is not allowed on a production server (Sandbox:AllowOnProduction).

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

|

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.
Sandbox failed	If the server does not require a sandbox the request is not blocked; the result is visible in the detail and in the report and the approver is expected to see it. If the server requires a sandbox, a failed or stale sandbox blocks execution and scheduling.
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.

A COMMIT in the script or a COMMIT inside a procedure also makes data permanent, and this cannot always be seen in the script text. The product tracks the transaction id in the Oracle sandbox as well: if a data change becomes permanent during the trial, the trial stops at once, the result is recorded as failed and a "SANDBOX WARNING" line appears in the result text. In that case the target should be checked; the evidence file also shows this trial as an exception.

On an Oracle request containing DDL, DCL or TRUNCATE the sandbox attempt is blocked when started, and the statement types responsible are listed in the message.

Only Oracle scripts that manipulate data (INSERT, UPDATE, DELETE, MERGE, SELECT) can be sandboxed; if they call a procedure that commits, the monitoring described above applies.

On SQL Server and PostgreSQL targets DDL statements also run inside the transaction and are rolled back at the end of the attempt.

14.5 Sandbox request and window events

When a sandbox is requested, the event `ChangeRequest.SandboxRequested` (`ReleasePackage.SandboxRequested` for a package) is written in the name of the requesting user; the event is in the same transaction in which the sandbox record is written. After the Worker connects to the target and before the script runs it writes `SandboxStarted`, and after the rollback it writes `SandboxFinished` (outcome `rolledBack` or `failed`). The connection is opened with the name `Sandbox` or `SandboxRelease`. If the start event cannot be written the script does not run and the sandbox record is closed as failed. In Oracle the DDL block returns before a connection is opened, so no window is opened in that case either.

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 <code>MaxQueryTimeout</code> parameter.
Status	Preparing, queued, executed.
Success	Whether the execution succeeded.

15.3 Execution modes

Value	Mode	Behaviour
1	<code>AllOrNothing</code>	Every request runs inside a single transaction. If one fails, all are rolled back.
2	<code>StopOnError</code>	Each request runs in its own transaction. On an error the package

Value	Mode	Behaviour
		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.

Topic	Single request	Release 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.

While the ITSM connection is on, every request in the package is checked against its own ITSM record; the package has no ticket of its own. The checks made when the Execute button is pressed and when a package is saved with a planned time are described in Section 22.9. While the ITSM connection is off, the package runs without asking ITSM.

Segregation of duties applies to packages as well. The person who starts the package is the one who presses the Execute button or who saves the package with a planned time. The same rule as the Execute button of a single request applies to this person: if the organisation rule does not allow it, a requester cannot start a package that contains their own request, and an approver cannot start a package that contains a request they approved. The rule is applied when a package is saved with a planned time and on the Execute button; the worker checks the same rule once more before it starts the package. Every blocked attempt is written to the audit log as ChangeRequest.ExecuteDenied. A package without a planned time is not an execution decision, so preparing one is free.

The evidence dossier of a request that ran in a package has a Release package section: the package, the execution mode, the order of the request, the other requests in the package, the person who started the package, the package outcome and the package audit trail. When a package is saved with a planned time, this decision is frozen for every request in the package together with the session of the decision maker (Section 18.8).

15.7 Window in package execution

In AllOrNothing mode the package uses a single connection and opens a single window (ReleasePackage.ExecutionStarted and ExecutionEnded; at the start the request numbers in the package are in the requestIds field and their tickets, in the same order, in the changeTicket field). In this mode which command belongs to which request is read from the audit trail and the evidence dossier, not from the DAM record. In StopOnError and ContinueOnError modes each request opens its own window and the events carry packageId.

Section 16. Object History (DDL History)

16.1 What it is for

It shows which request changed a database object, when, by whom and what was done. The list is built from the applied requests inside SQL Change Guard: every row corresponds to a request and to the script of that request. Building the list does not connect to the target server. When the Compare with server button is used on a row, the product connects to the server at that moment and fetches the current definition of the object there.

This feature depends on the InsightsReporting module and requires the same permission as the Reports screen (report.view). Because the screen shows script bodies, the scope is restricted as well: a user without the permission to see all requests (cr.seeall) only sees records from their own requests, and the screen states this.

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	Servers named in applied requests. If a request targeted several servers, each server appears on its own row.
Database list	Databases holding objects that were changed on the selected server.

Step	What it shows
Object summary	Object name, schema, type (table, view, procedure, function, trigger, index and others), in how many requests it was changed and the last change. The search box runs on the objects of the database you expand.
Object history	Versions of the selected object: version number, request number, ticket, the action performed (CREATE, ALTER, DROP), statement type, who changed it and when it was applied. The version number follows the order of application and is the same for every user; for a user who can only see their own requests the numbers appear with gaps, because the versions they cannot see are still counted. Every row shows the first lines of the script; the View script button opens the full script of that version. Clicking two rows in turn places the two records side by side. The compare button at the end of a row compares that version with the current definition of the object on the server. The Current definition on the server button in the header shows the present definition of the object without selecting a version. The full script is shown from the text ledger of the approved save. If the current text of the script in the table differs from the approved text, the row shows the Table text differs mark and the script window shows a warning; this means the script was changed outside the application. The same difference is written as SCRIPT_TABLE_DIFFERS in the evidence dossier of the request.

16.3 Audit value

This screen gathers the past changes of an object made through the product in one place: the auditor question "when was this procedure last changed, under which ticket and by whom" is answered on a single screen, with the script itself right there.

Only applied requests enter the list: executed, marked as manually applied, or failed during execution on an Oracle target. The last case carries a Partial badge, because the change cannot be said to have been applied in full. Requests marked as manually applied carry a Manual badge: the product did not run the change, it was declared.

A change made outside the product does not appear as its own row in the list, because every row here rests on a request. The way to see such a change is to look at the definition of the object on the server: the Current definition on the server button in the header opens that definition directly, and the compare button at the end of a row compares it with the selected version. If the last applied version and the definition on the server differ, this indicates that the object was changed outside the product; because a

table definition is regenerated from the server, a difference in formatting alone can also appear, so the difference should be reviewed.

The same comparison is available on the request entry screen (Section 11.3). The product knows who made a change and when only for requests that pass through it. For a change made outside the product, what it can show is that the definition on the server does not match the record; who made that change and when is read from the target database's own tracking mechanism, which your database administrator sets up. The Object Change Frequency and Table Structure Changes reports are also used in this review.

Comparing two records does not mean the same thing for every object type. For a procedure, view, function or trigger every version carries the full body of the object, so the difference between two versions can be read directly. For a table the records usually carry the change that was made: one may be a CREATE TABLE and the other an ALTER TABLE ADD COLUMN. Those two records are not two states of the object, and the difference between them does not mean the object changed shape. The product states this in the comparison window; for the present state of the table the Current definition on the server button is used.

Section 17. Audit Trail and Integrity

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. The time is given by the database server; the application cannot send it.

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. The table below lists the main events the product produces; it is not a complete list. Execution and sandbox window events (Section 17.14), approval chain blocking events (ChangeRequest.ApprovalBlocked, ChangeRequest.ExecutionBlocked), reconciliation events (ChangeRequest.ReconciliationRequired, ChangeRequest.ReconciliationResolved) and evidence boundary events (Evidence.BoundaryBroken, Evidence.BoundaryRestored) are described in their own sections. The event name filter on the audit trail screen matches exactly, so names must be searched as written here.

Event	Meaning
ChangeRequest.Create	A request was created.

Event	Meaning
ChangeRequest.Approve	A request was approved.
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.
ChangeRequest.Update	The request was updated.
ChangeRequest.ApprovalStepApproved	An approval step was approved.
ChangeRequest.ApproveDenied	An approval attempt was blocked by the separation of duties rule.
ChangeRequest.ExecuteDenied	An execution attempt was blocked by the separation of duties rule.
ChangeRequest.QueuedForExecution	The request entered the execution queue; the person who pressed the button appears in this record.
ChangeRequest.AutoExecute	The request was started by automatic execution.
ChangeRequest.ExecutionFailed	Execution failed; depending on RejectOnError the request is rejected (installation value) or stays approved.
ChangeRequest.CancelExecutionRequested	Cancellation was requested for a running request.
ChangeRequest.ExecutionCancelledBeforeStart	A queued request was cancelled before the worker started it.
ChangeRequest.Schedule	The request was scheduled for a later time.
ChangeRequest.CancelSchedule	A scheduled execution was cancelled.
ChangeRequest.ScheduleClearedOnError	The schedule was cleared because of an error.
ChangeRequest.MarkManual	The request was marked as applied manually outside the product.
ChangeRequest.NoLongerNeeded	The request was closed as no longer needed.

Event	Meaning
ChangeRequest.AutoRejected	The request was rejected automatically.
ChangeRequest.RollbackGenerationFailed	The automatic rollback script could not be produced; the reason is in the record.
ChangeRequest.RollbackInvalidatedByEdit	The existing rollback plan was invalidated because the script was edited.
ChangeRequest.ParseErrorsConfirmed	Parse errors were confirmed by the user and the request was saved.
ChangeRequest.EmergencyReviewed	An emergency change was reviewed after the event.
ChangeRequest.RuleSnapshot	The rule set affecting the request was frozen at save time.
ChangeRequest.QrFileExpired	The query result file was deleted because its retention period ended.
Server.Create	A server definition was added.
Server.Update	A server definition was updated.
Server.Delete	A server definition was deleted.
Server.Activated	The server was activated.
Server.Deactivated	The server was deactivated.
Server.RollbackRequirementDisabled	Control weakened: the rollback requirement was switched off for the server.
Server.RollbackRequirementEnabled	The rollback requirement was switched on for the server.
Server.SandboxRequirementDisabled	Control weakened: the sandbox requirement was switched off for the server.
Server.SandboxRequirementEnabled	The sandbox requirement was switched on for the server.
Server.MaskingRelaxed	Control weakened: the masking regime was relaxed for the server.
Server.MaskingTightened	The masking regime was tightened for the server.
Server.AutoExecutionEnabled	Automatic execution for change requests was enabled on the server.
Server.AutoExecutionDisabled	Automatic execution for change requests was disabled on the server.
Server.QrAutoExecutionEnabled	Automatic execution for query requests was enabled on the server.
Server.QrAutoExecutionDisabled	Automatic execution for query requests was disabled on the server.

Event	Meaning
Parameter.Upsert	A parameter was added or updated.
Parameter.Delete	A parameter was deleted.
Parameter.ControlChanged	Control weakened: a control parameter was changed.
Parameter.DualControlDisabledNoApprover	Four eyes was switched off because no second approver was available.
Parameter.DualControlChangedNoApprover	The four eyes scope was changed because no second approver was available.
CriticalObject.Create	A critical object definition was added.
CriticalObject.Update	A critical object definition was updated.
CriticalObject.Delete	Control weakened: a critical object definition was deleted.
SqlStandard.Create	A SQL standard was added.
SqlStandard.Update	A SQL standard was updated.
SqlStandard.Delete	Control weakened: a SQL standard was deleted.
QrSensitiveColumn.Create	A sensitive column rule was added.
QrSensitiveColumn.Update	A sensitive column rule was updated.
QrSensitiveColumn.Delete	Control weakened: a sensitive column rule was deleted.
ServerApprovalPolicy.Upsert	A server to approval policy mapping was added or updated.
ServerApprovalPolicy.Delete	Control weakened: a server to approval policy mapping was deleted.
ReleasePackage.Create	A release package was created.
ReleasePackage.Update	A release package was updated.
ReleasePackage.Delete	A release package was deleted.
ReleasePackage.QueuedForExecution	A release package entered the execution queue.
License.Activated	A new licence was installed.
License.Reloaded	The licence was reloaded.
WorkerConfig.Delivered	Worker configuration was delivered.
WorkerConfig.AccessDenied	An unauthorised attempt to reach the worker configuration endpoint.
Ledger.DigestPublished	A ledger digest was produced; if an external store is configured (Evidence:Ledger:DigestOutputPath) it was written as a file.

Event	Meaning
TableWriteLog.CoverageRecorded	The daily scope summary of the database level write log was written to the ledger.

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 Ledger Status

The Ledger Status button on the Audit Log screen shows the current state of the evidence ledger. Permission: audit.view.

Field	Meaning
Evidence table is an append-only ledger	Whether the table is created as a SQL Server ledger table.

Field	Meaning
Last ledger digest	The time and block number of the last digest published.
On time	Whether the last digest is within the digest interval multiplied by Evidence:Ledger:DigestStaleAfterIntervals.
Written to the external store	Whether the last digest was written as a file to the external store.
External store configured	Whether Evidence:Ledger:DigestOutputPath is set.
Last restore	The last date of a Ledger.RestoreDetected event or of the database restore history.
Last fork	The last date of a Ledger.ForkDetected event: the database went back behind a digest written outside.
Critical findings and limitations	The findings of the evidence write boundary check (Section 17.16).

The result is one of three states. Healthy: the table is a ledger, the boundary is intact, the last digest arrived on time and was written to the external store. Attention: the boundary is intact but the digest is late, the external store is not configured, or a restore or fork was recorded in the last 30 days. Broken: the table is not a ledger or the boundary check found a critical finding; in that case state changing operations already stop.

The screen does not say "verified". That the records have not changed is shown by `sys.sp_verify_database_ledger` run on a copy where the backup is restored, with the digest files from the external store of the organisation. This verification is not done on the production server; the steps are in the Auditor Kit document.

Verify Script Integrity

Checks the approval chain of a specific request: it shows that the save, approval and execution events carry the same fingerprint, that the signatures are valid and that the approved texts are in the text ledger. If the current text in the table differs from the approved text, it reports this as a hint. Run from the request detail. Details: Section 11.6, Approval chain.

17.7 Scope summary of the write log

TableWriteLogCoverageJob runs inside the worker once a day (Workers:TableWriteLogCoverageSeconds) and writes the scope summary of the database level write log (row count, smallest and largest id, first and last record time) to the evidence ledger as a TableWriteLog.CoverageRecorded event. Because the event sits in the ledger, the database engine rejects changing it: if today's row count or id range does not match the previous event, rows were deleted from the write log (Section 5.5).

The evidence ledger itself does not need this job: the protection of the ledger, the ledger digest and the evidence write boundary check are described in Sections 17.16 and 17.17.

17.8 Delivering the evidence outside the organisation

The external witness of the ledger leaves the organisation in two ways: the signed digest files are written to the external store of the organisation (Section 17.17) and the digest event goes to the SOC when SIEM is on (Section 17.12). The auditor takes the digest files of the period from that store, and the requests and control health of the period from the period package (Section 17.18).

The period package is downloaded with a separate permission (audit.anchorexport, "Can Export Period Package" on the roles screen). This permission is enabled by default for the Auditor and SystemManager roles. It is separate from the evidence dossier permission because of segregation of duties: the evidence dossier permission is enabled by default for the DbAdmin role as well, and DbAdmin is the role most likely to be the subject of an evidence investigation. The organisation may grant this permission to the DbAdmin role too if it wishes.

17.9 Timestamping (optional)

A signature shows which installation a file came from, not when it was produced. That date comes from the organisation's own clock, and an organisation can set its clock back. Timestamping closes this gap: the digest of the file is signed by a timestamp authority, either outside the organisation or one the organisation runs itself.

This feature is off by default. The product runs on premises and timestamping is an optional addition. As long as Evidence:Timestamp:Mode is Off, no network call is made at all.

Two settings are enough to turn it on: set Evidence:Timestamp:Mode to On and put the address of the timestamp server into Evidence:Timestamp:Url. The organisation may use its own internal server; from the product's point of view there is no difference between an internal and an external one. Proxy address and credentials can also be defined if needed.

A separate mode can be given for the evidence dossier (DossierMode). It can only narrow the main setting, never open it. If Mode is Off, no timestamp is obtained. The reason is that an operator who wants to close outbound access must be able to rely on a single setting.

Organisation data does not travel to the timestamp server. What is timestamped is the digest of the file, not the file itself. So even when an external authority is used, no content leaves the organisation.

If a timestamp cannot be obtained, nothing stops. If the server is unreachable, the time limit is exceeded or the reply is malformed, the file is produced without a timestamp and the flow continues. The gap is not hidden: no timestamp file (Manifest.tsr) is placed into the evidence dossier, and a warning is written to the application log.

Whether timestamps are actually being obtained is visible on the Governance Readiness screen. When timestamping is on, a line appears there showing the result of the last timestamp attempt since the API started; if there has been no attempt since startup, the row is shown for information. If it is red the most likely causes are: outbound access is blocked, the proxy requires credentials, the address is wrong, or the certificate cannot be validated because the proxy opens and re-signs the traffic.

If outbound access goes through a proxy, enter its address, user name and password into the proxy fields under Evidence:Timestamp. The proxy password is a secret like any other; it is expected to be held in an environment variable, and the readiness screen warns if it sits in the configuration file. If the proxy address is left empty, the proxy defined by the operating system is used; that does not mean no proxy is involved.

The timestamp lies outside the area covered by the signature. This means the removal of a timestamp from a package cannot be detected. This is a deliberate limit and an acceptable one: removing a timestamp weakens the evidence, it does not create false evidence.

Corporate proxies usually expect the service account Windows identity rather than a user name and password. In that case enable Evidence:Timestamp:ProxyUseDefaultCredentials; you then do not need to define a password at all. This option only works when the application runs under a domain account. If you enter a user name explicitly, that one is used.

If the proxy expects a domain identity and the application does not run under a domain account, fill in Evidence:Timestamp:ProxyDomain next to the user name. Putting the domain in front of the user name works with most proxies, but it does not hold for all of them; a separate field removes that uncertainty. Leave the field empty if the proxy does not ask for a domain. On the screen and in the log the proxy summary appears as DOMAIN\user; the password appears nowhere.

If the organisation timestamp server is on the internal network, enter its address into Evidence:Timestamp:ProxyBypass so that requests do not go through the proxy. Several addresses are separated by semicolons and wildcards are allowed, for example: tsa.company.local;*.company.local. When a proxy is defined, local addresses are bypassed already, so you do not need to list them.

To try the connection, use Settings > Diagnostics > Timestamp Test. The upper part of the screen shows the settings the application uses, read only, including which proxy is actually in effect. In the lower part you can enter temporary values and try them; what you type there is never saved and your settings do not change. The proxy password is never returned to the screen, only whether one is set.

The test ignores the timestamping mode, so you can try the connection while it is still off. The reason is that having to switch a setting on before you can try it would mean trying it in production.

If the attempt fails, the reason is named. This matters because each reason belongs to a different team. Proxy required credentials and proxy could not be reached point at the system and network teams, certificate could not be validated points at the security team, and timestamp server could not be reached points at outbound access rules. Reply could not be verified usually means a proxy sign-in page came in between.

The most common cause of certificate could not be validated is a proxy that opens encrypted traffic and re-signs it with its own certificate. On such a network the connection cannot be validated even when the timestamp server is reachable. The fix is not inside the product: the organisation root certificate has to be present in the trust store of the machine that runs the application. On Windows that store is Trusted Root Certification Authorities under Local Computer. If the application and the worker are on separate machines, the certificate has to be installed on both. The product offers no setting that turns certificate validation off; a timestamp taken over an unvalidated connection would remove the assurance the timestamp is there to give.

17.10 The Audit Log screen

Column	What it shows
Time	Event time, in your browser's local time.
Action	The event name.
Entity Type	The affected record type.
Entity Id	The affected record number.
Performed By	The user or system actor.

Column	What it shows
IP	The request address.
Old Value / New Value	Shown side by side when the row is expanded.

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

17.11 The audit records package: delivering the raw records

The evidence dossier tells the story of one request; the period package gives every request and the control health of the period. There is a third delivery: the raw records. The auditor uses it to examine the events of a range in their own tool.

Screen: Audit Log. Permission: audit.view. How to use it: press the Evidence Package button and enter the first and last record number. The auditor does not have to guess the range; the evidence dossier states it (Section 18.3).

The package is a ZIP file and uses the same wrapper as the evidence dossier and the period package. Package.json is the signed content: the records in the range (event, entity type, entity id, old and new value, performed by, address and time). Manifest.json carries the package header and the digest of every file. Manifest.sig is the signature of the manifest. PublicKey.pem is the public key. Verify.ps1 is the verification script. All three deliveries are verified with the same procedure and the same script.

The package fixes the content at delivery time with a signature. That the records have not changed afterwards is shown not by the package but by the ledger: the same records are in the ledger, and the ledger is verified with the digests in the external store (Section 5.4).

The export itself is audited: the action is written to the audit trail as EvidencePackageExported. Because the same event name is also used for the evidence dossier, the record also states which delivery was taken.

17.12 SOC integration (SIEM)

The audit trail lives inside the product. In an organisation, security events are watched at the SOC, and the team there does not look at the product's screens. This integration feeds governance events into the organisation's own collector, so that database side activity is watched in the same place as everything else.

When events are sent

Delivery is done by a poll that runs every ten seconds by default. On each round the poll reads the audit records written after the delivery marker and sends them; when there is no new record it sends nothing. With the default setting an event is therefore usually sent within ten seconds; the time to reach the collector depends on the network and the collector. The interval is changed with the `Workers:SiemPumpSeconds` key.

Delivery is done by the worker, not the API. The reason is that there must be a single sender: the API can run as more than one copy, and if each copy sent on its own the order would break and duplicates would increase.

Delivery resumes where it stopped

The product keeps a delivery marker for each endpoint and each stream: sent up to here. The order is fixed; the batch is sent first and the marker is moved afterwards. If delivery fails the marker does not move and the next round continues from the same place.

Even if the collector stays down for three days no event is lost on the product side. When it comes back the accumulated events flow in order. In return, a crash can cause the last batch to be sent again; in a stream that carries evidence, a duplicate is always better than a loss. This assurance covers the sending side of the product: over UDP the product cannot confirm that the collector received the message, so a setup that expects lossless delivery should use TCP or TLS. Sending an event does not by itself show that the SIEM received or processed it.

No separate queue is kept on disk. Log shippers on the market write events to disk when the collector is unreachable, because they hold no other copy of the data. Here the situation is different: the event already sits in the append-only ledger table, and the database engine rejects deleting it. Building a second queue would mean holding the same data in two places and creating the chance for the two to drift apart.

The one exception is the product's own log stream. That stream is held in memory and when the queue fills up the oldest record is dropped; during an incident the most valuable records are the newest ones. The dropped count appears in the heartbeat message and on the Governance Setup Check screen. There is no loss in the audit trail and ledger digest streams.

Heartbeat

A heartbeat message is sent every five minutes. It carries the last sent record number, the delivery lag, the number of dropped log records and the uptime. Its purpose is to let the SOC tell two kinds of silence apart: no events happening, and the product having stopped. The SOC can raise an alert when the heartbeat stops.

The heartbeat is not affected by the severity threshold. It is sent even when the threshold is set to the highest level; otherwise an organisation raising the threshold would, without realising it, close the only way of knowing that the product has stopped. The legitimate way to turn it off is to remove the Heartbeat stream from the SiemStreams parameter.

What is sent and what is not

The event identity, name, time, who performed it, source address, related request number, server and outcome are sent. Only named fields leave the event payload; everything else is dropped by default. This is why an allow list is used instead of a deny list: with a deny list, a new field added tomorrow would flow to the SOC without anyone noticing.

Values that look like passwords, keys and similar secrets are masked as well. Script text is not sent by default; the script digest goes instead.

Query result data is never sent under any condition. What reaches the SOC is the fact that the query ran, not the result itself.

17.13 How to read the message that reaches the SOC

The SOC team does not look at the product's screens; it looks at the line that lands in the collector. This section explains how that line is read. It carries what is needed by whoever sits with the SOC engineer during installation.

A sample message

The default format is an RFC 5424 syslog header with a JSON body. A real message looks like this:

```
<133>1 2026-09-20T07:08:22.128Z SERVER01 SQL_Change_Guard 13544 146 -  
{ "eventId":146, "stream":"Audit", "action":"ChangeRequest.Create", "entityType":"ChangeRequest", "entityId":"812", "user":"jane.doe", "severity":"Notice", "outcome":"success", "occurredAt":"2026-09-20T07:08:22.1280000Z" }
```

The header fields are separated by spaces and their order is fixed:

Field	What it means
<133>	Priority number. A single number that encodes the facility together with the severity; explained under the next heading.
1	Syslog version. Always sent as 1.
2026-09-20T07:08:22.128Z	The time of the event. The trailing Z says the value is UTC. Time is always sent in UTC because the most common problem in SOC correlation is a time zone difference.
SERVER01	The name of the machine that sent the message.
SQL_Change_Guard	The application name. Used in the collector to tell the source apart.
13544	Process id. When more than one copy of the product runs, this field separates them.
146	Message id. It is the row number of the audit record, and the collector uses this field to drop duplicates.
-	Structured data field. Not used in this format, so the nil marker is sent.
{ ... }	Body. The JSON that carries the fields of the event. Fields without a value are not written at all.

The message id field carries a different number in each stream. In the audit trail and ledger digest streams it is the audit record number and it never changes, so in those two streams the collector can safely drop duplicates. In the product's own log stream it is a counter kept inside the process and it restarts from zero when the process restarts. In the heartbeat and in the scope announcement it is the timestamp of the moment the message was produced.

The priority number and local0

In syslog every message starts with a single number, and that number carries two pieces of information at once: which source class the message came from, and how serious it is. The source class is called the facility and the degree of seriousness is called the severity.

The facility is the numbered class syslog uses to separate sources roughly. Values from zero to twenty three are defined; the first numbers are reserved for the operating system's own components, while local0 through local7 are left to the organisation's own applications. The product uses local0 by default and the value is changed with the `Siem:Endpoints[].Facility` key. If another application in the collector also uses local0, an unused facility such as local3 can be given to the product to keep the two apart. If an unknown facility name is written the message is still sent; local0 is assumed.

The priority number comes from this formula: $\text{priority} = \text{facility number} \times 8 + \text{severity number}$. The number of local0 is 16.

Example: an ordinary audit record has the severity Notice, and the syslog number of Notice is 5. $16 \times 8 + 5 = 133$. That is the <133> at the start of the message above.

The five severities the product uses and the priority values they produce with local0:

Severity	Syslog number	Priority with local0	When the product uses it
Alert	1	129	Emergency handling, permission denial and control weakening events
Error	3	131	A failed operation
Warning	4	132	Unmasked data output, evidence package export and application warnings
Notice	5	133	Ordinary audit records and ledger digest publication
Info	6	134	Heartbeat

The product does not use the two highest syslog degrees (Emergency and Critical). Those degrees are meant for cases such as the system itself becoming unusable; putting a governance event in that class would lead to wrong prioritisation on SOC dashboards.

This number can be used directly when writing alert rules in the SOC. A common setup is this: turn every <129> event into a case immediately, review <131> and <132> events during the day, only store and search <133> events, and raise an alert when the <134> heartbeat stops.

How the severity is decided

The severity comes from the name of the event; there is no separately entered importance field on the audit record. Instead of a fixed list the rule works on the name pattern: every event whose name ends with Denied counts as a permission denial, and every event whose name ends with Failed or Rejected counts as a failure. This way no list update is needed when a new event is added to the product, and no unrecognised event disappears silently; an unrecognised event falls to the safe default, which is Notice.

Examples with real event names:

Event	Which rule it matches	Severity
ChangeRequest.Create	No rule, default	Notice
ChangeRequest.Approve	No rule, default	Notice
ChangeRequest.Executed	No rule, default	Notice
Ledger.DigestPublished	Ledger digest	Notice
ChangeRequest.ExecutionFailed	Name ends with Failed	Error
ChangeRequest.Reject	Failure	Error
QrZipPasswordView	Data exposure	Warning
EvidencePackageExported	Data exposure	Warning
Evidence.PeriodPackageExported	Data exposure	Warning
LoginFailed	Permission denial	Alert
ChangeRequest.ExecuteDenied	Name ends with Denied	Alert
ChangeRequest.EmergencyReviewed	Emergency handling	Alert
User.Delete	Identity control change	Alert
Parameter.ControlChanged	Control weakening	Alert
Siem.FilterChanged	Control weakening	Alert

Event	Which rule it matches	Severity
SsoLoginFailed	Permission denial	Alert
LoginBreakGlass	Emergency handling	Alert
SsoRoleMapping.Update	Control weakening	Alert
User.SyncedFromIdp	Identity control change	Alert
LdapLoginDenied	Name ends with Denied	Alert
User.CreatedFromDirectory	Identity control change	Alert
User.SyncedFromDirectory	Identity control change	Alert
User.DisabledInDirectory	Identity control change	Alert
Ldap.ConfigurationChanged	Identity control change	Alert
Ldap.SyncAborted	Identity control change	Alert
Ledger.ForkDetected	Ledger continuity broken	Alert

An event can match more than one rule, and in that case the highest degree wins. For example `SettingChange.Rejected` is both a settings change record and a rejection record. Because the control weakening rule is applied first, the event is sent as Alert and not as Error.

A user changing their own password (`User.PasswordChanged`) is an ordinary operation and carries the Notice degree. An administrator resetting a password (`User.AdminPasswordReset`) counts as a control change and is sent with the Alert degree.

The rule is different in the product's own log stream: there is no event name there, so the level of the log line is mapped directly. Fatal becomes Alert, Error becomes Error and Warning becomes Warning; the remaining lines are sent with the Notice degree.

The message that reaches the SOC when the scope is narrowed

When one of the four parameters described in Section 7.11 changes, the product sends a separate event. The event is sent with the configuration that was in force before the change took effect, and it carries the old value together with the new one. A real example:

```
<129>1 2026-09-20T18:28:43.089Z SERVER01 SQL_Change_Guard 19696 1789928923089 -
{"eventId":1789928923089,"stream":"Audit","action":"Siem.FilterChanged","entityType":"Para
meter","entityId":"SiemStreams","user":"admin","severity":"Alert","outcome":"success","oldVa
lue":"Audit;Anchor;AppLog;Heartbeat","newValue":"Anchor;AppLog;Heartbeat"}
```

The body written out in readable form:

```
{  
  "eventId": 1789928923089,  
  "stream": "Audit",  
  "action": "Siem.FilterChanged",  
  "entityType": "Parameter",  
  "entityId": "SiemStreams",  
  "user": "admin",  
  "severity": "Alert",  
  "outcome": "success",  
  "oldValue": "Audit;Anchor;AppLog;Heartbeat",  
  "newValue": "Anchor;AppLog;Heartbeat"  
}
```

In this example the Audit stream has been taken out of scope. Audit is present in oldValue and absent from newValue; the difference shows directly which stream was closed.

The field to look for in the SOC is action, with the value Siem.FilterChanged. This event itself is affected neither by the severity threshold nor by the stream selection; the fact that a stream was closed is sent under all conditions. Otherwise an organisation could close the Audit stream first and then make every later narrowing silently.

A recommended alert rule: turn every event whose action field is Siem.FilterChanged into a case in the SOC. This event appears a few times a year and produces no noise.

17.14 Execution and sandbox window events

Each interval in which the product does work on a target is called a window. Each window is written to the audit trail as a "started" and an "ended" event. A session in the DAM record that falls inside this interval, uses the account of the product and comes from the expected address is the work of the product.

Event	Written by	Meaning
ChangeRequest.ExecutionStarted and ExecutionEnded	Worker (WorkerService)	The interval in which a request ran on the target. In a query result request, a separate window for each server
ReleasePackage.ExecutionStarted and ExecutionEnded	Worker	AllOrNothing packages only (single connection)
ChangeRequest.SandboxRequested	API, on behalf of the requesting user	A sandbox was requested. The target has not been contacted yet
ChangeRequest.SandboxStarted and SandboxFinished	Worker	The interval of the sandbox trial
ReleasePackage.SandboxRequested, SandboxStarted, SandboxFinished	API and Worker	The same for the package sandbox

The payload of the start event contains these fields: windowKind (kind of work), serverName, executionLogin (the account the product uses on the target, never the password), executorHost (name of the Worker machine), clientAddress (address seen by the target), targetSessionId (session number), targetAuditSessionId (Oracle only), sessionApplicationName (the name given to the connection), executionStartedAt (UTC), and where relevant packageId, requestIds (requests in the package, separated by semicolons), sandboxId, changeTicket (the ITSM record of the request, for example CHG0030001) and emergencyTicket (the incident record of an emergency request, for example INC0012345).

The payload of the end event contains windowId, windowKind, windowOutcome, serverName, executionFinishedAt (UTC) and where relevant packageId and sandboxId. windowId is the audit record number of the start event. The end event does not repeat the account, the address and the session; it is tied to the start by windowId.

The ticket fields let the SOC see which ITSM record an opened window belongs to without going to the request number. Rules: the tickets are only in the start event (the end event is tied to the start by windowId); the value is the ticket frozen when the request was saved, it is not read again from ITSM at that moment; for a request without a ticket the field is not written at all; for an emergency request without a change record only emergencyTicket is present; a rollback request carries the ticket of its source request; in a query result request every server window carries the same ticket; in an AllOrNothing package changeTicket and emergencyTicket are a semicolon separated list in the same order as requestIds and a request without a ticket stays as an empty element (for example requestIds 5;6;7 and changeTicket CHG0030005;;CHG0030007); sandbox windows do not carry tickets. Outcome (windowOutcome)	When
executed	The target committed the change. For a query result, the read from the server succeeded
failed	The script failed and was rolled back on the target. For a query result, a read error or the row limit was exceeded
cancelled	The user cancelled, or the work was cut short when the service shut down
rolledBack	The sandbox script ran successfully and was rolled back
abandoned	The process crashed and the end could not be written. The window was closed afterwards; the real outcome is in the request, package or sandbox record

Rules:

- If the start event cannot be written, the script does not run at all. The request is not rejected; it returns to Approved. A failure of the audit database is not a fault of the request. For packages, the next requests are not started.

- The start event is written in its own short transaction before any transaction is opened on the target and on the application database; even if the execution fails, the "started" proof is not lost.
- If the end event cannot be written, the result of the work does not change and the error is written to the Worker log. The next orphan close completes the missing end.
- If a user cancels a request while it is waiting, before the Worker has connected to the target, no window is ever opened; the request returns straight to Approved (ChangeRequest.ExecutionCancelledBeforeStart). If it is cancelled while the script runs, the window is closed with cancelled.
- These events have Notice severity and travel in the Audit stream. The ChangeRequest.Executed and ChangeRequest.ExecutionFailed events are written separately from the window events.

A query result request opens a separate Query window for each target server. The connection is opened with the name SQL Change Guard/Query/<request number>, the start event is written before a transaction is opened on the target, the read is done inside a transaction and the transaction is rolled back without a commit (no change stays on the target). The end event is written when the read from that server ends: executed if the read succeeded, failed on a read error or when the row limit is exceeded, cancelled if cancelled. In this way a read (SELECT) record in DAM that has the product account, the same session number and falls in the time range of the window can be tied to the query result request it belongs to. The Health, Preview, Definition and Test connections of the product also read but do not open a window.

17.15 Carrying the window events to the SIEM

The fields of the window events (windowId, windowKind, windowOutcome, executionLogin, executorHost, clientAddress, targetSessionId, targetAuditSessionId, sessionApplicationName, executionStartedAt, executionFinishedAt, packageId, requestIds, sandboxId, changeTicket, emergencyTicket) leave for the SOC on purpose: the account name, the machine name and the client address are needed for the SIEM to answer the question "does this session fall into a window of the product". The password and the connection string are never in the payload. The fields are carried in all formats (JSON, RFC 5424, LEEF, CEF). Because CEF has limited fields, window events are written to fixed fields:

Attribute	CEF field
executionStartedAt, executionFinishedAt	start, end (epoch milliseconds)
executionLogin	duser
executorHost	shost
clientAddress	src (if the event's own address is empty; it is empty for Worker events)
sessionApplicationName	requestClientApplication

Attribute	CEF field
windowId	cn1 (label windowId)
targetSessionId	cn2 (label targetSessionId)
sandboxId, otherwise packageId	cn3 (the label is whichever it is)
serverName, windowKind, windowOutcome	cs4, cs5, cs6
targetAuditSessionId	Not carried in CEF. Present in JSON, RFC 5424 and LEEF
changeTicket	cs3 (label changeTicket)
emergencyTicket	flexString1 (label emergencyTicket)
requestIds	flexString2 (label requestIds)

For an organisation that uses Oracle, a format other than CEF is recommended. Fields are cut at 256 characters on the way to the SIEM; in a very large package requestIds and the ticket list may be cut, and the full list is in the audit trail and the evidence dossier. If the account name contains a word such as token, secret or password, it is masked on the way to the SIEM; the product account must not be given such a name. The host field in the SIEM message is the Worker machine that sends the event, not the target server.

17.16 Evidence ledger and evidence write boundary

Audit records are kept in the AuditLogs table of the evidence schema in the application database. The table is a SQL Server append-only ledger table: the database engine rejects changing or deleting records for everyone, sysadmin included. The product writes to the table only through the evidence.usp_Audit_Append procedure; the procedure takes the event time from the database server clock itself (UTC), so the application cannot write a back-dated or future-dated record. Reports and screens read the dbo.AuditLogs view. This protection is about the content of the table; on its own it is not enough against returning the database to an older backup or changing the database files directly. Those cases are detected with the ledger digest written to the external store (Section 17.17).

The ledger protects what the procedure writes; it does not protect that the procedure writes correctly. This is why the product checks the following boundary at API start-up, every 15 minutes and with the Worker's own job: is the evidence table a ledger, does it have a trigger, is the write procedure the version the product released, is there an unexpected object in the evidence schema, does another object reference the table, was the table dropped and recreated, is TRUSTWORTHY or DB_CHAINING on.

If the boundary breaks, the product writes the Evidence.BoundaryBroken event, sends a notification email and stops state changing operations such as saving, approving, rejecting and

executing requests until the boundary is restored; screens stay readable. When it is restored, Evidence.BoundaryRestored is written. If the evidence table is missing, is not a ledger or the database is older than SQL Server 2022, the API does not start at all.

An event that changes the state of work (saving a request, approval, rejection, execution result) is always written in the same transaction as that change. If the event cannot be written, the change is not made either. On a failed sign-in, the name typed by the user is not recorded if the account does not exist in the system; "unknown" is written, because users sometimes type their password into this field and it cannot be deleted from the ledger. The content of an event is scanned before it is recorded: password, secret and key fields are masked; a valid Turkish identity number, card number or IBAN in free text such as a description or reason is masked, and the names of the masked fields are written to the _redacted list of the event.

17.17 Ledger digest

Once an hour (Evidence:Ledger:DigestIntervalMinutes) the Worker starts a digest round; the API takes a ledger digest from the database. The digest is a single value that covers every ledger record up to that moment. The API signs the digest (when signing is on), writes it as a file to the external store configured with Evidence:Ledger:DigestOutputPath (with a .sig file next to it when signing is on) and adds the Ledger.DigestPublished event to the audit trail; if SIEM is on, the event also goes to the SOC. If no external store is configured, no file is written. Making the store immutable (for example WORM or a retention lock) and keeping database administrators from writing to it is the organisation's responsibility. When these conditions are met, even if someone who manages the database changes records later, the digest written outside does not change, so the change shows up in the auditor's verification.

If a digest cannot be produced or written to the store, Ledger.DigestFailed is written. If the database is restored, the API notices it from the msdb restore history during the digest round and writes Ledger.RestoreDetected. The "Last ledger digest" row on the Governance Readiness screen warns if no digest arrives for two intervals; the same rule is set up in the SOC.

Continuity check: in every digest round the latest digest file in the store is read and compared with the ledger chain of the database (the "previous block hash" of the next block must equal the digest written outside). If the database is behind that digest (return to an older backup) or the chain does not match (restore on another server, a change in the database file), the Ledger.ForkDetected event is written and a critical notification is sent. This check does not re-hash any rows, so it runs in production in every round; it catches a fork even if the restore history is deleted, as long as a digest file exists in the external store.

The "10b. Evidence State" section of the evidence dossier shows the external digest that covers the last record of the request and states one of three states: "Covered by an external digest",

"Not yet covered" or "Verification failed". The "12. Field Verification" section verifies the ledger with `sys.sp_verify_database_ledger` while the dossier is generated and writes the result: VERIFIED (the external digest file in the immutable store was read and the ledger was verified up to it), VERIFIED_INTERNAL (the external file could not be read and only the digest inside the ledger was used; this is not an independent verification and the dossier does not say "verified"), NOT_COVERED (no digest covers the last record yet), UNAVAILABLE or FAILED. The product's own verification is not independent; the auditor gives the independent verdict by running the same procedure on a copy of the database with the digests from the immutable store. The steps are in the Auditor Kit document.

17.18 Period package

Screen: Audit Log. Permission: `audit.anchorexport` ("Can Export Period Package" on the roles screen; the segregation of duties reason is in Section 17.8).

How to use: on the Audit Log screen select the start and end date and press the Period Package button. The period is at most 92 days; one package per month is the recommended use. The package is delivered to internal audit and its receipt is recorded in the document system of the organisation.

The package contains `Population.json` (every request of the period reconciled row by row between the request table and the audit trail; the same content as the Period Population report), `ControlHealth.json` (ledger digests published in the period, gaps between digests, failed digests, boundary broken and restored events, restores, worker outages and the result of today's boundary check), `Exports.json` (the log of evidence files and packages taken in the period), `Chain.json` (the approval chain status of each request and the attempts blocked at the chain gate in the period), `Manifest.json`, `Manifest.sig`, `PublicKey.pem` and `Verify.ps1`. The export itself is written to the audit trail as `Evidence.PeriodPackageExported`. If an event type has more than 20,000 records the package is not truncated; it is not produced at all, and the period must be shortened.

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, ITSM checks, sessions of the decision makers, audit trail and the audit record range of the request. The package digest is computed over this file.
Manifest.json	The package header: format name and version, installation id, product version, request number, production time, the user who produced the file, the id of the signing key, the package digest (packageHash), the audit record id, and the SHA-256 digest of every file in the package.
Dossier.pdf	The same information in readable, printable form.
Manifest.sig	The signature of Manifest.json. Produced with the organisation's private key. The auditor verifies it with the public key and sees that the file came from this installation. Included only while signing is enabled.
PublicKey.pem	The public key used to verify the signature. It is not confidential and can be shared. It can verify a signature but cannot produce a new one. Included only while signing is enabled.

The request's audit trail is included in full, up to 100,000 records. If that limit is exceeded the trail is truncated and this is written into the file as an exception.

The file also states the audit record range of the request: the first and the last record number. If the auditor wants to examine the events of that range in raw form, they ask for the audit records package of the same range (Section 17.11). Without the range the auditor would have to ask the organisation which package to request; letting the audited party define the scope of its own evidence is a weak arrangement for an audit.

Some of the numbers inside the range do not belong to this request, and that is normal. The audit trail is the log of the whole organisation; while one request is being handled, other users act as well. A request's rows are therefore not contiguous.

The file list inside Manifest.json lets the auditor check every file in the package one by one. If a single character of a file changes, its digest no longer matches.

The package also contains the Verify.ps1 verification script (Section 18.6). When time stamping is enabled, the RFC 3161 time stamp of the manifest is added as Manifest.tsr.

The installation id (installationId) sits inside the signed content. If a signed evidence file of one organisation is placed into another organisation's package, the installation id does not match and verification reveals it.

Signing is a setting of the organisation (Evidence:Signing:Enabled). If signing is off, the file is produced and Manifest.sig and PublicKey.pem are not placed inside it; it can still be shown that the package has not changed since it was produced, but not which installation it came from, and this is visible from the content of the file. If signing is on and the key cannot be used (no key defined, certificate not found, private key cannot be read), the file is NOT produced and the user sees a "could not be signed" message: a file that is expected to be signed is never silently produced unsigned.

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.

Integrity of the masking record: at delivery time the digest of each server's masking record is written to the evidence ledger (QueryResult.MaskingRecorded event). When the dossier is produced, the stored record is digested again with the same rule and compared with the value in the ledger. The result is printed under each delivery in the file: verified if the record matches the digest written at delivery, changed if it does not (the verdict of the file turns red), and no trace if the ledger holds no record of it. Trust in the masking information in the file therefore rests not only on the signature of the file but also on the ledger event written at delivery time.

The file also shows how many columns and rows were delivered in each delivery. The lists of masked and unmasked columns contain sensitive columns only. If a column requested unmasked is not in the result or is not sensitive anyway, that row is written as an exemption with no effect; it is not written as delivered unmasked.

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.

The event stays in the audit trail; it does not enter the timeline inside the file. The reason: if downloading the file entered the file itself, a request that has not changed at all would produce a different package digest on every download. As it works today, an unchanged request gives the same package digest however many times it is downloaded, so an auditor can compare two files byte for byte. Nothing is lost: how many times evidence was produced for a request is visible on the Audit Log screen and in the Data Access and Export Log report.

The digest of Manifest.json differs on every download, and that is expected: the manifest carries the production time, the user who produced the file and the audit record id. Comparison is always made on the package digest (packageHash).

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.

Field	Meaning
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. That the records the content rests on have not changed is shown by the evidence ledger (Sections 5.4 and 17.16).

Verifying without touching the system

The path above requires the auditor to reach the product. There is a second path that requires no access at all: the signature inside the package is verified with the public key that is also inside the package. The auditor does this on their own computer with a common tool. No confidential value of the organisation is shared.

1. The ZIP file is opened.
2. The SHA-256 digest of every file listed in Manifest.json is computed and compared with the listed value. This shows that no file in the package has changed.
3. The Manifest.sig signature is verified against Manifest.json using the PublicKey.pem public key. Example command: `openssl dgst -sha256 -verify PublicKey.pem -signature Manifest.sig Manifest.json`

Expected output: Verified OK. If both steps pass, two things are shown together: the package is exactly as it was when produced, and at that moment it was signed with the key of this installation.

One point deserves attention: the auditor must separately confirm that the public key they hold really belongs to the organisation. Otherwise the organisation could hand over a package signed with a different key and present it as its own. The way to confirm this is to obtain the fingerprint of the key from the organisation through a separate channel. The fingerprint is the SHA-256 digest of the binary form of the public key, not the digest of the PEM file itself.

The organisation reads this fingerprint from the Governance Readiness screen under Settings. The signing key fingerprint row shows the exact value the auditor will compare against. Hand that value to the auditor through a channel separate from the package: a letter, a contract annex or a corporate portal. The product cannot see whether this was done, so the row stands as a reminder.

The verification script inside the package

Every package contains a verification script named `Verify.ps1`. The auditor can unzip the package and check it with a single command: `powershell -ExecutionPolicy Bypass -File Verify.ps1`

The script can also be run from Windows Explorer: right-click the file and choose Run with PowerShell. A window opened this way prints the result and then waits for Enter, so the output stays on screen. When the script is called from another script, add the `-NoPause` parameter and it does not wait. If an unexpected error occurs the script does not close silently: it prints the error and states that the error says nothing about the package itself.

The script checks the digest of every file listed in the manifest, verifies the manifest signature and prints the fingerprint of the public key. Because it reads what to check from the manifest, the same script works for the evidence dossier, the period package and the audit records package.

The digest of the script is written into the manifest, and the manifest is signed. A later swap of the script therefore shows up when the manifest signature and the script digest are checked with standard tools (`openssl`, `Get-FileHash`); the "passed" output of a swapped script is not evidence of anything.

The script is not part of the evidence and says so in its own header. Everything it does can also be done by hand. An auditor who wants independence should run the same checks with their own tools at least once; a tool that ships inside the package it checks is not proof on its own when it reports a pass.

The script does not report success when it could not verify the signature. Signature verification also works on the PowerShell version that ships with Windows, so the auditor does not need to install anything. If the key cannot be loaded for any reason, the script reports the result as

incomplete, states which commands can be run instead, and returns a different exit code. Matching file digests do not mean the signature was verified.

18.7 Very long scripts and the PDF

When a request carries many long scripts the PDF grows to thousands of pages and turns into a document nobody reads. To prevent this, script text inside the PDF is shortened when needed. A note is placed where the text was cut, stating how many lines were hidden and where the full text can be found.

Shortening applies only when it is needed. In an ordinary request with a single script, a few hundred lines are written out as they are. As the number of scripts grows, the space available to each one shrinks.

Shortening does not affect evidential value. The full text of the scripts is always inside Dossier.json, and the package digest is computed over that file. The PDF exists to be read; verification is done over Dossier.json.

18.8 Sessions of the decision makers

For every human decision on a request, the identity details of the decision maker's session at that moment are frozen together with the decision. The evidence dossier shows them in the "Sessions of the Decision Makers" section.

Field	Meaning
Decision	Request saved, approval step (with its number), rejected, execution requested, execution scheduled, execution requested with a release package, release package saved with a planned time, emergency change reviewed, marked as applied manually, no longer needed.
Login	Local password, corporate directory (LDAP) or corporate account (SSO, with the provider name).
MFA	Yes (the product's own code), Yes (reported by the identity provider), No, or Not reported by the identity provider.

Field	Meaning
Emergency login	A session opened with a local password while corporate sign-in is on. A decision made with such a session is also written to the Exceptions list.
Session number	The same number as in the login record in the Audit Log (LoginSuccess, MfaSuccess, SsoLoginSuccess, LoginBreakGlass). The auditor links the decision to the login with this number.

The details are written at the moment of the decision and are not recalculated later. Even if the person's account is later converted to a corporate account or they sign in another way, what the dossier shows is the session of that day. For a session without evidence (opened before these details were kept), the login shows as "Unknown"; nothing is made up.

The system's own actions (scheduled execution, automatic approval) are not in this section because there is no personal session; they appear in the timeline as SYSTEM or WorkerService.

18.9 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, the finding that set it and the 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.
Did this file really come from this organisation?	The manifest signature and the public key.

Question	The answer in the file
How had the decision maker signed in at that moment?	Sessions of the decision makers: login method (local password, corporate directory, corporate account), MFA, emergency login and session number (Section 18.8).
Was there a permission in ITSM?	ITSM checks: the state seen in ITSM on save and on execution, and the decision made (Section 22).
Did it run in a release package, and who started the package?	Release package: the package, the execution mode, the order, the other requests in the package, the person who started the package (pressed the Execute button or saved the package with a planned time), the package outcome and the package audit trail.

18.10 Execution windows and exceptions

The evidence dossier contains an Execution windows section. The start and end events in the audit trail of the request and in the audit trail of the AllOrNothing packages that contain the request are joined by windowId. For each window the window number, kind, server, start, end and outcome are written; beneath them the account, machine, address seen by the target, session number and application name. It is also present in Dossier.json as the executionWindows field. Query result data does not enter the file; the window carries information only.

If there is more than one end for the same window (for example, when the recovery threshold is passed, first abandoned and then the real end of the work is written), the last end is valid. Exceptions are produced from the last end.

Exception code	Meaning
EXECUTION_CANCELLED	The execution was cancelled
EXECUTION_ABANDONED	The execution window was left unfinished (the process stopped); read the real outcome from the request record
SANDBOX_ABANDONED	The sandbox window was left unfinished
EXECUTION_WINDOW_OPEN	It has a start but no end, and the request is not currently Executing

With this section the auditor answers from inside the file the question "with which account, from which address, with which session and in which interval did this request run on the target", and matches the DAM record with this interval.

Two more exception codes are produced in addition to the table.

EXECUTION_LOGIN_MISMATCH: the account the target saw in the window differs from the account in the server definition; DAM reconciliation is done with the observed account.

SANDBOX_TARGET_COMMITTED: the sandbox window closed with the executed outcome, meaning data became permanent on the target during the trial (for example a COMMIT in the script or a procedure on Oracle).

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	Approved requests scheduled within the next 24 hours from now.
Pending settings approvals	Settings changes awaiting a decision.

19.4 Execution success rate

The formula is $\text{successful} / (\text{successful} + \text{failed})$.

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.
Unapproved unmasking	Requests where unmasked columns were requested but not approved.
ITSM ticket rule violated	Request attempts with a closed, missing or wrong type ITSM record (rejected ones and those passed with a warning).

Counter	What it counts
ITSM verification not possible	Verifications passed under the rule because ITSM could not be reached.
Execution stopped by the ITSM gate	Executions stopped because the ITSM record was not ready.
Passed without the CAB check (rollback, emergency)	Rollback and emergency requests that passed the execution gate without waiting for CAB approval.

The card also contains these rows: High or Critical band requests approved by rule without any person approving them, requests with an approval or execution attempt blocked by segregation of duties (Section 27.4), unreviewed emergency changes (Section 27.5) and requests awaiting reconciliation (ReconciliationRequired). The last two counters are not affected by the date range. If any of these is higher than expected, the detail is in the Control Exceptions report. The detail of the ITSM rows is in the ITSM 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.

19.9 The explanation marks on the dashboard

Every number on the dashboard has a small information mark next to it. Hovering over it tells you what the number counts, which time range it looks at and where the detail can be seen. You therefore do not need to open this guide to answer "where does this number come from".

For example, the mark in the most active users section says the ranking is by the number of requests created, not by who executed or approved them. Every row under control exceptions has its own mark that names the screen or the report holding the detail.

Rows whose value is zero are shown in the control exceptions list as well. Zero means it never happened in that period, not that the control was removed; hiding the row would give the wrong impression. Every row is clickable and opens the screen holding the detail: per request details open in the Control Exceptions report (with the date range from the dashboard already filled in), blocked segregation of duties attempts open on the Audit Log screen, and emergency changes awaiting review open in the request list. The question of which records are behind a number is therefore one click away.

Clicking a number opens the request list showing exactly those records. The filters you saved earlier (date range, only my requests, favourites) are not applied while opening. Otherwise the dashboard would say four while the list showed one row, and the two screens would contradict each other.

19.10 The worker indicator and outage records

The worker service reports its own status to the API every 30 seconds. This report is called a heartbeat. If 90 seconds pass without one, the worker is considered stopped.

The status is shown in the top bar of the application, in the same place on every screen:

- While the worker runs, a green dot and the word Worker are shown. Hovering displays the time of the last report and how the mechanism works.
- When the worker has stopped, a red bordered warning reading "Worker stopped" is shown. Hovering displays the time of the last contact, what it means and what to do.

The warning means this: scheduled work and requests waiting for execution do not progress until the worker is back. Requests are not lost, they wait; when the worker returns, work continues where it left off. The indicator refreshes every 30 seconds and the warning disappears on its own when the worker returns.

The indicator lives in the top bar, not on a single screen. There is no telling which screen you will be on when the problem appears, so the information is in the same place everywhere.

Outages are written to the audit log

The indicator in the top bar tells you the current state, not the history. The auditor asks about the past: why did nothing run last night, how many hours was the worker down.

For that reason an event is written to the audit log whenever the state changes:

- Worker.Offline: the worker stopped reporting. The record holds the time of the last report and how long ago it was.
- Worker.Online: the worker is back. The record holds how many minutes the outage lasted.

These records are not written by a user; they are performed by field reads SYSTEM. Records are produced only on a state change. Writing one on every check would have filled the audit log with a row every 30 seconds and made it unreadable.

When the application has just started, the worker may not have reported yet. That is not an outage and produces no record; monitoring begins after the first real report.

How to check: stop the worker service, wait at most two minutes and see the indicator in the top bar turn red. Find the Worker.Offline event on the Audit Log screen. When you start the service again the indicator recovers on its own and a Worker.Online event is written, holding how many minutes the outage lasted.

Section 20. Reports

Screen: Reports

Permission: report.view

Module: InsightsReporting

20.1 General rules

- Reports are organised into five groups and there are 34 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.
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?"	-

Report	What it shows	The question it answers	Restriction
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.	"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 as SYSTEM. If it ran inside a release package, the package is shown too.	"What ran in production?"	-
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. Request sandboxes and release package sandboxes are in the same list; the Scope column tells them apart. A package row lists the requests in the package separated by ';', and its risk band is the highest band of	"Was it tried before going live?"	-

Report	What it shows	The question it answers	Restriction
	those requests. An outdated test does not cover the current scripts.		
Control Exceptions	Requests that left the normal flow: emergency records, overridden steps, self approved steps, unmasked data requests without approval, blocked segregation of duties attempts (how many attempts were made and who made them, separated by ';'), approvals and executions stopped at the approval chain gate (the content, the plan or the governance state did not match), executions in a release package whose composition did not match, and high risk requests approved by rule without any person approving them. A request cannot be saved for a server without an approval policy, so requests without a policy do not appear in this list.	"Where were the controls bypassed?"	Auditor, DbAdmin, ChangeManager, SystemManager
ITSM Control Exceptions	Every moment the ITSM check left the normal flow: ticket rule violated, verification not possible, execution stopped, request passed without the CAB check. Each row shows the state seen in ITSM at that moment and the decision.	"Where did the ITSM check stop work, and where was it skipped?"	Auditor, DbAdmin, ChangeManager, SystemManager
Period Population	Every request of the period, collected separately from the request table and the audit trail and reconciled row by row. Each row shows the status in the table next to the status derived from the audit trail; the PopulationCheck column	"How do I know I received every request of this period?"	Auditor, DbAdmin, ChangeManager, SystemManager

Report	What it shows	The question it answers	Restriction
	states Match, NotInAuditTrail, NotInTable, StatusMismatch or InProgress. Rows that do not match come first.		

Auditors usually start with the Control Exceptions report. If it is empty, the product recorded no control exception in the selected period; operations performed outside the product do not appear in this report. The fourteenth report of the group is Emergency Change Review: it lists emergency changes with their reason, incident ticket and post event review status, with rows awaiting review first (Section 27.5). Restriction: Auditor, DbAdmin, ChangeManager, SystemManager.

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, MERGE and procedure calls, with the last execution date.	Tables receiving frequent manual data fixes point to a process problem.
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, authentication type, linked corporate identity providers, last corporate sign-in 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. Corporate sign-in events are listed too: accounts opened by the identity provider, roles synchronised from groups, identity links, group to role mapping changes, every break-glass sign-in (local password sign-in while corporate sign-in is on) and every change of the corporate sign-in settings.	Auditor, DbAdmin, ChangeManager, SystemManager

The Expected Access Principals report

Expected Access Principals: shows with which account, from which machine and from which address the product connects to each target server. The "expected source" list on the DAM and SIEM side is built from this report. Report permission: Auditor, DbAdmin, ChangeManager, SystemManager.

Filters: date range (default last 30 days) and server name. One row comes for each registered server; a server with no work is also listed.

Column	Meaning
ServerName, Environment, DatabaseType	Server name, its environment and its type (SqlServer, PostgreSQL, Oracle)

Column	Meaning
ProductLogin	The SQL account the product uses on that server. The password is not in the report
SeenRoles	The kinds of work seen in the range (Execute, Query, Release, Sandbox, SandboxRelease), separated by semicolons
SeenHosts	The machine names the product connected from, separated by semicolons
SeenClientAddresses	The client addresses seen by the target, separated by semicolons
LastUsedAt, WindowCount	Time of last use and the number of windows in the range

Limit: the list comes only from work that opens a window (the Worker). The connections of the API machine (preview, object definition read, connection test) do not open a window; the address of the API machine does not appear in this report and must be found separately. The report is evidence, so it does not do dirty reads, and it does not contain the account password.

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
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 and rows were delivered (DeliveredColumnCount, DeliveredRowCount), how many sensitive columns were masked, how many sensitive columns were delivered unmasked and whether that was approved. The masked and unmasked	Auditor, DbAdmin, ChangeManager, SystemManager

Report	What it shows	Restriction
	counts cover sensitive columns only; a delivery without sensitive columns shows zero for both. If an approved unmasked request opened no column (the name is not in the result or the column is not sensitive anyway), it appears in the UnmaskRequestNoEffectCount column. Deliveries made while masking was off appear with the MaskingOff policy.	
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, with old and new values. Changes made on screen: servers, parameters, approval policies, roles, SQL standards, critical objects, sensitive columns, patterns, mail sources, ITSM tickets, license, corporate identity role mappings and maintenance tasks. Changes made in the settings file	Auditor, DbAdmin, ChangeManager, SystemManager

Report	What it shows	Restriction
	(record type Configuration): application settings, corporate sign in, directory and ITSM connection settings. App.ControlRelaxed marks a settings change that relaxed a control. The AuditLogId column links every row to the audit trail event.	
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 Integration and Ticket Number

22.1 What it is for

It links the number of the record in the organisation's ITSM system (change record, service request, incident record) to the request, so that the SQL Change Guard record matches the corporate process record. While the ITSM integration is on, the number is verified in ITSM and the work done is written to the ITSM record as a note (Section 22.5).

Screen: Settings > ITSM

Permission: settings.itsm

22.2 Fields

Field	What it does
Record no	The record number. Number and kind together are unique.
Title	The subject of the ticket.
Link	An address to open the ticket directly.
Source	Who wrote the row: Manual, From request, or the name of the ITSM product (ServiceNow, Jira, GenericRest).
Kind	Change record, incident record or service request.
ITSM state	The state last seen in ITSM and the product's class (Ready, Awaiting approval, Closed, Not classified, Open). Filled only while ITSM is on.
Planned window	Planned start and end in ITSM. Filled only while ITSM is on.
Last verified	When the record was last queried in ITSM.

22.3 Requirement and format validation

The change ticket is governed by two parameters. The incident number asked on emergency requests has its own parameters: EmergencyTicketRequired and EmergencyTicketRegex. All four are described field by field in Section 7.6.

Parameter	Effect
ChangeTicketRequired	Sets on which requests a ticket number is mandatory: Always (all of them), ExceptEmergency (all but emergency ones), Never (none).
ChangeTicketRegex	Validates the format of the entered number. No check when empty.

Parameter	Effect
RequestTicketRegex	Validates the format of the number when a data extraction request is based on a service request. When empty, ChangeTicketRegex applies.

Example: with ChangeTicketRequired set to Always 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.

22.4 When a row is added to the ticket list

When a request is saved, the number entered on it is added to the ITSM records list if it is not already there, so that the same number can be picked from the list on later requests. While ITSM is off, the source of the row is From request. While ITSM is on, the row is written with the details seen in ITSM and its source is the name of the ITSM product (Section 22.12). Which request added the row and who did it is written in the audit log (`ItsmTicket.AutoCreated`).

The order is as follows: first all checks on the request run, then the ticket row is added, then the request is saved. This has three practical consequences.

1. A request that fails a check leaves no ticket behind. If saving is rejected because a deactivated server was selected, for example, no new row appears in the ITSM records list. Otherwise the list would fill up with numbers from requests that were never saved.
2. A number that fails format validation never reaches the list. A number that does not match `ChangeTicketRegex` is rejected on save and no row is created.
3. Requests carrying the same ticket do not wait for each other. Every request in a release package may carry the same ticket number; because the ticket row is added outside the save operation and held only briefly, requests do not queue up behind one another. If two people add the same number at the same moment, the list still keeps a single row and neither of them gets an error.

How to check: with ITSM off, open Settings > ITSM. Save a request with a number that is not in the list and see that the number is added. Save a second request with the same number and see that the list still shows a single row. Try saving with a number that does not match the format: the request is not saved and no row appears in the list.

The link between the ticket row and the request is one way: the request uses whatever number it carries. Deleting a row from the list does not affect requests carrying that number.

22.5 ITSM integration

SQL Change Guard can connect to the organisation's ITSM system. Supported systems: ServiceNow, Jira (Cloud and Data Center) and other ITSM products that expose their records through a REST API (such as ServiceCore). While the connection is on, the product does three things:

1. When a request is saved, it looks the number up in ITSM: does the record exist, is it the right kind, is it closed.

2. Right before execution, it looks the record up again: has the change record been approved by the CAB and is it in a state where it can be implemented.

3. It writes the important steps of the request (created, approved, scheduled, executed, failed, stopped by the ITSM check) to the ITSM record as notes.

The product does not open records in ITSM, does not change their state and never sends script text, result data or passwords to ITSM. The state seen in ITSM when a decision is made is written to the audit log; even if the ITSM record changes later, what was seen at that moment is not lost.

While ITSM is off (ItsmEnabled = false), the product never contacts ITSM; the number is checked only by the format and requirement rules (Section 22.3).

The approval in ITSM and the approval in the product answer different questions. The CAB approval in ITSM answers "should this work be done". The approval in the product answers "should this script run on this server in this form". Both are needed.

22.6 Record kinds

Kind	Example	Where it is used
Change record	CHG0030001	The "Change record no" field of a change request
Service request	RITM0010001	The "Service request no" field of a data extraction request
Incident record	INC0012345	The "Incident no" field of an emergency request

While ITSM is on, the name of the field on the request screen follows the kind of the request; while ITSM is off, the field shows as "Ticket no" on every request. Which record a data extraction request is based on is set by ItsmQueryResultTicketKind: Request (service request), Change (change record) or None (not verified in ITSM, the field shows as "Ticket no"). The kind is written to the request when it is saved; even if the parameter changes later, older requests are looked up with their own kind.

22.7 How the state in ITSM is read

Because state names differ between organisations, the product maps the state of a change record to one of four classes:

Class	Meaning	How it is decided
Ready	Can be executed	The state is in ItsmReadyStates and, when ITSM keeps the approval in a separate field, the approval is in ItsmApprovedValues
Awaiting approval	Open but not ready yet	The state is not in the ready list or the approval has not been given yet
Closed	Closed or cancelled	The state is in ItsmClosedStates. In Jira, every state in the "Done" category
Not classified	Not counted as ready	The state or the expected approval field did not arrive

In ServiceNow the CAB approval is a field separate from the state. The product looks at both: even if the state is Implement, the record is not ready unless the approval is Approved. In Jira the approval is expressed by the workflow state, so only the state is read.

Incident records and service requests have no CAB approval. They show as Open or Closed.

22.8 When a request is saved

When you leave the number field, a coloured label next to it shows the state of the record in ITSM, and the title of the record appears below it. The button next to it opens the record in ITSM. The label is for information only; the decision is made on the server when you save.

ITSM answer	ItsmVerifyOnSave = Block	ItsmVerifyOnSave = Warn
Record exists, right kind, not closed (even without CAB approval yet)	Saved	Saved
Closed, not found or wrong kind	Not saved	Saved and recorded as an exception
ITSM cannot be reached	According to ItsmUnavailablePolicy	According to ItsmUnavailablePolicy

CAB approval is not required on save: a request can be prepared before the CAB meets. The approval is required before execution. A rollback request linked to a source is never rejected on save. On an emergency request the incident number is looked up in the same way, following `ItsmVerifyIncident`.

22.9 Right before execution

With `ItsmExecuteGate = Block`, the record is looked up in ITSM again on every execution path (Execute button, scheduled execution, automatic execution after approval, release package). This lookup does not use a cache. The Execute button checks twice: when the button is pressed and right before the worker starts the script, because the approval may have been withdrawn in between. The Execute button of a release package works the same way: when it is pressed, every request in the package is checked against its own ITSM record. If a request stops, the package is not queued and all stopped requests are listed in one message.

Situation	Result
Change record Ready	Runs
Awaiting approval or Not classified	Stops
Closed, not found or wrong kind	Stops
With <code>ItsmEnforceWindow = true</code> : outside the planned window, or no window on the record	Stops
Data extraction based on a service request that is not closed	Runs

A stopped request does not become dead; it stays Approved and can be executed again when the ITSM record is ready. If a scheduled execution stops, the schedule is cleared and an e-mail goes to the requester and the approvers. If one request in a release package stops, the package does not start at all, so it never ends up half applied.

The record is also looked up in ITSM when a request is scheduled with the Schedule button or when a release package is saved with a planned time. If the request would certainly stop at that time, the schedule is refused: the record is closed, not found or of the wrong type, or, with `ItsmEnforceWindow = true`, the scheduled time is outside the planned window. If the CAB approval is not there yet, the record has no planned window or ITSM cannot be reached, the schedule is accepted, because these can change before that time; the decision is made at execution. A refused schedule is written to the audit log and to Control Exceptions and is not sent to the ITSM record as a note. Emergency requests and rollbacks linked to a source are exempt from this check as well.

In these three cases the schedule is saved but the user is warned at that moment: a warning window opens on screen and stays until it is closed. Example: if a request is scheduled for 02:00 at night while the change record is in Authorize (no CAB approval), the window says that the execution will stop if the record is not approved by then. If the CAB approves before 02:00, the request runs at that time. If not, the execution stops, the request waits as Approved, the schedule is cleared and an e-mail goes to the requester and the approvers. In a release package the warnings are shown one by one with the request number. The warning is not a decision and is not written to the audit log; the evidence is the ITSM decision at execution.

There are two exemptions. A rollback request linked to a source is exempt from this check: after the source change is applied, the ITSM record is usually closed, and reversing something that broke must not be blocked. On an emergency request the CAB approval is not required, because in most organisations an emergency change is approved afterwards; a closed, missing or wrong kind change record still stops an emergency request. Every exemption is written to the audit log and to Control Exceptions.

22.10 When ITSM cannot be reached

ITSM can go down, the network can drop or the password of the integration account can change. What happens is set by `ItsmUnavailablePolicy`: `AllowAndRecord` (default) lets the work continue and records that the verification could not be made; `Block` stops saving and execution. Emergency requests pass in both cases.

"Record not found" and "ITSM cannot be reached" are different: the first is a rule violation, the second a connection problem. An authentication error (wrong integration account or password) counts as not reachable and shows in red on the Governance Readiness screen.

The worker does not connect to ITSM; it asks the API for the decision before execution. If the API cannot be reached (network outage, API restarting), `ItsmUnavailablePolicy` applies. If the API answers but gives no decision (for example the API key on the worker is wrong or the API has an error), the execution is not started: this is a setup error, not ITSM being unreachable, and it does not open the gate. Emergency requests and rollbacks linked to a source pass in both cases. Both situations appear with their reason in the ITSM checks of the request.

22.11 Notes written to ITSM

Request events are written to the ITSM record as notes: a work note in ServiceNow, a comment in Jira. `ItsmWriteBackEnabled` switches notes on or off and `ItsmWriteBackEvents` sets which events are written. A note carries the request number, what happened, the server, who did it, the time, the risk band and a link to the request. Example:

SQL Change Guard #1042: executed successfully. Server: SQLPROD01. By: WorkerService (started by ayse.dba), 01.10.2026 22:06 Europe/Istanbul (UTC+03:00). Risk band: Medium. Details: <https://change-guard.company.local/change-requests/1042>

Notes are sent in order. If ITSM cannot be reached, notes are not lost; they continue from where they stopped once ITSM is back. If no note can be sent for 15 minutes, the `Itsm.WriteBackStalled` alarm is written once. If a note cannot be written because the record was deleted in ITSM, the note is skipped and this is recorded as well (`Itsm.WriteBackSkipped`).

22.12 The ITSM screen

At the top of Settings > ITSM is the connection card: which ITSM the product is connected to, the result of the last lookup and the state of the notes. The Test connection button looks up a sample record number in ITSM and shows the state seen, the class, the approval and the planned window. The test also works while ITSM is off; use it to verify the settings before switching the integration on.

Below it, the ITSM records list works in two ways. While ITSM is off, the list serves the suggestions on the request screen and is managed by hand; numbers typed on requests are added here as well. While ITSM is on, the list is a copy of ITSM and is read only: every record verified on save or on execution is added with its title, state and planned window. A row added by hand would suggest a permission that does not exist in ITSM.

The ITSM checks panel on the request detail screen shows every ITSM check made for that request: time, stage, decision, state in ITSM, approval and reason. The same information is in the evidence dossier.

22.13 Setup

The connection details are in the `Itsm` section of `appsettings.json`: Provider (ServiceNow, Jira or GenericRest), `BaseUrl`, `AuthType`, Username and Password or Token. The password and the token are secrets and are kept in environment variables in production (`Itsm__Password`, `Itsm__Token`). Every change in this section is written to the audit log with the `Itsm.ConfigurationChanged` event when the API starts.

Give the integration account the least permission: read records and add notes. Do not give it permission to change states or delete. The API server needs HTTPS access to the ITSM address; no connection from ITSM to the product is needed. The worker does not connect to ITSM.

Suggested order:

1. Define the connection in `appsettings.json` and restart the API.

2. On Settings > ITSM, run Test connection with a real record number.
3. Set ItsmReadyStates, ItsmClosedStates and ItsmApprovedValues to your organisation's state names together with your ITSM administrator.
4. Set the ticket format parameters (ChangeTicketRegex, RequestTicketRegex, EmergencyTicketRegex) to match your numbers.
5. Set ItsmEnabled to true.
6. Check the ITSM integration group on the Governance Readiness screen.

22.14 Points to watch

- All ITSM parameters are control settings: a change is written to the audit log and, if settings approval is on, waits for a second administrator.
- If a state before CAB approval is added to ItsmReadyStates, an unapproved change can be executed. Decide the list together with your ITSM administrator.
- The four ITSM rows on the dashboard and the ITSM Control Exceptions report show violated ticket rules, verifications that could not be made, stopped executions and requests that passed without the CAB check. The morning review of emergency changes is done with this list.

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. Audit records need 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 difficult.

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 refresh tokens, used or expired password reset tokens, consumed or expired sign-in tickets and corporate sign-in requests.	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.
Governance Readiness	Compares post-installation governance settings with the values expected in production. It holds nine groups; the number of rows depends on the features in use (corporate sign-in, directory, ITSM, SIEM and so on) and each row is read live. For every row and what it means see Section 27.8.	On a row showing Attention, go to the screen named in its Managed in column and correct the setting. On a row showing Decide there is nothing to correct; what is expected is a conscious choice by the organisation.

25.5 Worker monitoring

The worker service reports its own status to the API every 30 seconds; if 90 seconds pass without a report, the worker is considered stopped. The status is shown in the top bar of the application on every screen, and the start and the end of an outage are written to the audit log; the detail is in Section 19.10.

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.
External digest store	The ledger digest files (Evidence:Ledger:DigestOutputPath). They are the basis of independent verification; they are kept without deletion for the retention period of the organisation. Protection against deletion and change depends on the store the organisation chooses (a WORM share, an object locked store and so on); the product does not provide it.	Same as the main database.
Encryption keys	Security:ServerPasswordKey, Security:QrZipPasswordKey, Jwt:SigningKey and the evidence signing keys (Evidence:Signing:Keys; old public keys are never deleted).	Whenever they change.
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 decrypt the stored server passwords or the passwords of query result packages; if the private signing key is lost, new evidence cannot be signed. Old signed packages remain verifiable with the public key, so old public keys and their fingerprints must be kept. 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. A request that certainly did nothing on the target returns to Approved; a request that may have committed on the target is moved to ReconciliationRequired and is not executed again, and a person makes the reconciliation decision. 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.

Setting	Meaning
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.
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 executions shorter than 45 minutes are not affected by recovery.

If you do not want to wait for recovery you can cancel the execution on the stuck request (requires the `cr.execute` permission). Marking a request as manual is not available while it is Executing; it is available only in the Pending and Approved states.

Execution window during recovery

The open execution window of each recovered request is closed in the same round with the outcome abandoned. The threshold is at least 5 minutes; a value smaller than 5 is ignored and 120 minutes is used (a warning is written when the Worker starts). If a job that passes the threshold is still legitimately running, recovery counts it as abandoned and closes its window with abandoned; when the job finishes the real end is also written and the last end is valid. For this reason the threshold must be larger than the longest legitimate execution. After the value is changed, the Worker must be restarted.

25.8 Watching SIEM delivery

The Settings > Diagnostics screen has a SIEM Status panel. For each endpoint the panel shows whether it is on, when the last successful delivery happened, how many records the delivery lag is, how many consecutive failures were recorded and what the last error was.

The same panel has a Send Test Message button. The test message travels the same path as real events: the same formatter, the same transport, the same connection. A separate path would make it possible for the test to pass while real events never arrive. The test works even while the master switch is off; an organisation wants to see the connection working before turning the integration on.

The Governance Setup Check screen carries nine SIEM checks: is the integration on, is an endpoint defined, is the transport encrypted, is the delivery lag reasonable, was the last successful delivery recent, has the stream coverage been narrowed, has the severity threshold been raised, is script text delivery on, and are records being dropped from the product log queue.

Common situations

Symptom	Likely cause and what to do
The test message arrives but real events do not	The worker is not running. Check the state of the Windows service. The test message is sent by the API, while real events are carried by the worker.
The delivery lag keeps growing	The collector may be accepting the delivery but processing it slowly, or the batch size may be below the event rate. Raise BatchSize.
The connection is refused with a certificate error	TlsServerCertThumbprint does not match the collector certificate. In this case the product

Symptom	Likely cause and what to do
	does not fall back to unencrypted delivery; it refuses the connection.
Some events do not arrive	The SiemMinSeverity threshold may have been raised or SiemStreams may have been narrowed. Both changes are announced to the SOC as a separate event; look for that announcement at the collector.
Product log records are missing	Loss is possible in this stream and it is deliberate. Raise AppLogQueueSize, or raise AppLogMinLevel to reduce the volume.

25.9 Verifying the SOC connection without a collector

The organisation's SOC collector is not always ready during installation; the security team usually has a schedule of its own. A small listener is enough to see that the product sends correctly without waiting for it: a program that listens on a port and writes the incoming line to the screen.

The script below is run in a PowerShell 7 window and needs nothing else installed:

```
& {
    $listener = $null
    try {
        $listener = [System.Net.Sockets.TcpListener]::new([System.Net.IPAddress]::Loopback, 6514)
        $listener.Start()
        Write-Host "Listening: 127.0.0.1:6514 (press Ctrl+C to stop)" -ForegroundColor Green
        while ($true) {
            $client = $listener.AcceptTcpClient()
            $stream = $client.GetStream()
            $buffer = New-Object byte[] 65536
            while (($read = $stream.Read($buffer, 0, $buffer.Length)) -gt 0) {
```

```

        $msg = [System.Text.Encoding]::UTF8.GetString($buffer, 0, $read)

        Write-Host "Incoming message: $msg"

    }

    $client.Close()

}

}

finally {

    if ($null -ne $listener) {

        $listener.Stop()

        Write-Host "Listener stopped." -ForegroundColor Yellow

    }

}

}

```

Then set the syslog endpoint in appsettings.json to Host 127.0.0.1, Port 6514 and Protocol Tcp, and restart the API. Press the Send Test Message button on Settings > Diagnostics > SOC / SIEM. The raw syslog line appears in the listener window.

This exercise shows three things at once. First, the product really opens the connection. Second, it writes the message in the expected format; the SOC engineer can see the line and write their own parsing rule before the installation is even finished. Third, no record is lost while the collector is down.

To see the third one, follow this order: stop the listener with Ctrl+C, create a few requests in the product, watch the Last error column fill in and the delivery lag grow on the SOC / SIEM screen, then start the listener again. The accumulated records flow in order, the lag drops to zero and the Last error column is cleared.

The listener does not accept an encrypted connection; with Protocol set to Tls the handshake fails and a certificate error appears on the screen. This setup is only for installation testing. When moving to the organisation's collector, Host, Port and Protocol are filled in for the real collector and Protocol is set to Tls.

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 çalıştırılmaz.	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.		-

Message (EN)	Message (TR)	Cause	Fix
The package was not started: request #{0} did not pass the ITSM check. {1}	Paket başlatılmadı: #{0} numaralı talep ITSM kontrolünden geçemedi. {1}	The ITSM record of a request in the package is not ready for execution.	Fix the ITSM record according to the reason in the message or remove the request from the package. See Section 22.9.
The package could not be scheduled: request #{0} would not pass the ITSM check at the scheduled time. {1}	Paket zamanlanamadı: #{0} numaralı talep planlanan saatte ITSM kontrolünden geçemez. {1}	The record is closed or the planned time is outside the window in ITSM.	Choose a time inside the planned window in ITSM or remove the request from the package. See Section 22.9.
The ITSM check stopped the scheduling: {0}	ITSM kontrolü zamanlamayı durdurdu: {0}	The same check when a single request is scheduled.	Choose a time inside the planned window or fix the ITSM record.
Segregation of duties does not allow you to execute request {0} contained in the package.	Paketteki {0} numaralı talep için görevler ayrılığı kuralı çalıştırmanıza izin vermiyor.	The person who starts the package, or saves it with a planned time, is the requester or an approver of a request in the package.	Let another authorised person start the package or save it with a planned time.
The package was not started: segregation of duties does not allow {1}, who started the package, to execute request #{0}.	Paket başlatılmadı: görevler ayrılığı kuralı, paketi başlatan {1} kullanıcısının #{0} numaralı talebi çalıştırmasına izin vermiyor.	The worker applied the same rule in its check before starting the package.	Set the package up again with another authorised person.

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.

Message (EN)	Message (TR)	Cause	Fix
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 maskeleyim 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.
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.
Is it a corporate account	An account of the Corporate account type cannot use the password screen; use the "Sign in with ..." button. If you see a message asking you to contact your administrator after the return, your group is not mapped to a role or your account is linked to another identity; your administrator checks this on Settings > Corporate Identity (SSO and LDAP).

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.

Check	What to do
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.
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.

Check	What to do
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. Governance Controls and Transparency

This section describes the mechanisms that close the gap between a control existing and a control working. A rule may be written in the handbook; the real question is whether a breach is seen without anyone having to ask.

27.1 Missing governance step indicator

In the request list and on the request detail page, steps that should have been done but were not appear as a red badge. Hovering over the badge lists each gap. On the detail page the same information sits as a panel at the very top.

The badge covers six cases. The server requires a rollback plan but there is none. A rollback plan exists but its script is incomplete. Automatic rollback generation failed. A sandbox trial is required but was never run. The last trial failed. The trial passed but the script changed afterwards.

The badge appears only on live requests: pending approval, approved (awaiting execution), executing and awaiting reconciliation. Dead requests (executed, rejected, marked manual, no longer needed) are not flagged. The rules in force on the day a request closed may differ from today's; flagging a closed request with today's rules produces a retrospective and false accusation.

The badge appears only on change requests. A query request has no concept of rollback or sandbox trial; putting a badge there would fill the list with meaningless warnings and over time all badges would be ignored.

Screen and command give the same answer. On a request with gaps the execute and schedule buttons are disabled, and hovering over them states the same reason. For these gaps the screen never allows what the server rejects; the evaluation is done in one place and fed to both the screen and the command from the same source. Checks made at the moment of the action, such as the approval chain, ITSM and the evidence write boundary, can still stop it while the button is enabled (Section 11.5).

27.2 Mandatory sandbox trial

Settings > Servers has a Sandbox required option per server. When it is on, a change can only be executed if a current and successful sandbox trial exists. The server list shows this setting as a separate badge, so which servers are protected is readable at a glance.

Currency is decided like this: the trial result must have been produced after the last change to the request. If the script is edited after the trial, the trial is treated as stale and execution is

blocked again. This closes the most misleading case: a new script going to production on the strength of an old script's successful result.

The rule is the same on all three execution paths: manual execution, scheduled execution and release package. The rule lives in a single service; because it is not duplicated, the three paths do not drift apart. A package does not replace individual execution; the trial state is verified separately for every request in the package. The flag depends on the module: if the related module is not licensed the gate does not work even with the flag on, so governance readiness shows that the module is unlicensed instead of the flag value.

The default is off. Before turning the setting on, make sure a sandbox environment is defined for that server; otherwise no change can be executed on it. Turning the setting off is written to the audit trail under its own event name, so when and by whom the control was relaxed is visible in a single query.

For Oracle changes that produce an implicit commit, the sandbox trial runs only in safe scenarios. For those changes protection comes not from the trial but from the mandatory rollback plan.

27.3 Integrity of the rollback plan

While a rollback script is being generated, the object definition may not be readable from the target server: a network outage, a maintenance window, or a missing object. In those cases an incompleteness marker is placed in the script and that script no longer counts as a valid rollback.

If rollback is mandatory on the server, execution with a half-written script is blocked, the screen states the reason, and the badge in the request list shows the gap. Going to production with a half plan is more dangerous than going without one, because the team believes a plan exists and finds out otherwise during the incident.

Automatic rollback generation runs at approval time in a system context. It does not depend on the approving user holding the add-request permission. In an organization where segregation of duties is set up properly the approver cannot raise requests at all; tying generation to that permission would punish the organization that set the control up correctly.

If generation fails, the request does not quietly stay approved. The request stores the generation-failed flag, its timestamp and the error reason; the event is written to the audit trail on behalf of the system and is visible on screen. The flag is cleared by itself once a rollback plan is created.

Having nothing to generate is not a failure. For a read-only script, or one with no effect to undo, producing no plan is normal and is not marked as a failure.

A rollback script generated from a table definition includes the primary key, unique constraints, check constraints, foreign keys and indexes. Items outside that scope are stated in a single note line at the top of the script. That note comes from one text source for SQL Server, PostgreSQL and Oracle, so the three database types do not produce different warning texts.

27.4 Blocked segregation of duties attempts

When the segregation of duties parameters (`RequesterCanSelfApproveExecute`, `ApproverCanExecute`) are switched off and a user tries to approve or execute their own request, the system blocks the operation. In the installation default these parameters are on and nothing is blocked. The block is written to the audit trail under its own event name: an approval attempt is recorded as `ChangeRequest.ApproveDenied` and an execution attempt as `ChangeRequest.ExecuteDenied`. The record contains the request number, the reason for the block and the user who attempted it.

This is what the record is worth: a blocked attempt is proof the control works. Without such a record an auditor cannot tell whether the rule really blocks or was simply never tested. Repeated attempts are also a behavioural signal and feed the organization's own internal review.

A failure to write the audit record does not change the outcome. The block always stands; the write failure only reaches the application log. Producing evidence must never override the protection.

The records appear per request in the Control Exceptions report: how many attempts were made and who made them, in separate columns. This is an evidence report and it runs without any dirty read hint. The individual records are on the Audit Log screen. The action filter matches the name EXACTLY; typing part of a name returns nothing and makes it look as if no record exists. To search for both names at once, separate them with a semicolon:

`ChangeRequest.ApproveDenied;ChangeRequest.ExecuteDenied`. Clicking the Blocked segregation of duties attempts row on the dashboard opens that screen with the correct names already filled in.

27.5 Emergency change review

The emergency path shortens approval, it does not skip it: approval is taken before the event, and at least one step of an emergency policy must be non-skippable. In addition, every executed emergency change carries a post-incident review record.

The panel appears at the top of the request detail page. It shows the requester, the executor, the emergency justification and the incident ticket. A user with approval permission answers two questions (was the change genuinely urgent, was a follow-up request raised) and writes a mandatory note. The note cannot be left empty; a review record without a note carries no evidential value.

The review is not an approval and carries no sanction. Even if the answer is no, it could have been planned, the request is not reversed and nobody is warned; the answer is simply recorded. With a sanction attached people would stop answering honestly and the record would lose its value. The purpose is not punishment but making the pattern visible: emergency changes coming from the same team every month point to a planning problem, not a process one.

The constraints are these. The requester cannot review their own request. A review is possible only for applied requests. The situations that count as applied are: the change was executed, it was marked as manually applied, or it failed during execution on an Oracle target. The last one depends on the target database. On SQL Server and PostgreSQL targets every block of a request runs inside a single transaction; when an error occurs they are all rolled back and nothing from the request remains in production, so there is no event to review. On Oracle, DDL statements are committed implicitly, so blocks that ran before the error may remain in production, and that is the event most in need of a review. A review is done once and is not edited afterwards. Permission is verified both from the session token and from the live database, so a user whose permission has been revoked cannot review with a token still in hand.

The dashboard has an unreviewed emergency change counter; clicking it lists the pending requests directly. The counter is not affected by the selected date range, because a pending obligation is not a past event and must not disappear when the range is narrowed. The counter, the filter in the request list, the Not reviewed badge on the list and the button on the request detail page all use the same definition: emergency requests that were applied and have not been reviewed yet. Using one definition in all four places prevents work that is invisible on the dashboard from looking actionable on the request page, and prevents a request without a badge on the list from being counted on the dashboard. An emergency request that failed during execution therefore carries the badge as well.

The Emergency Change Review report lists emergency changes with their justifications, incident tickets and review status. Pending items come first with how many days they have been waiting. The record also enters the request's evidence dossier as its own section, so an auditor opening the evidence package sees there whether the review was done.

27.6 Effect of unmasked column approval on the risk band

When approval is given to show unmasked sensitive columns in a query result, the risk band of the request is recalculated. The script has not changed, but the request has genuinely become riskier; a band that does not reflect this misleads whoever reads it.

The recalculation is rebuilt from the stored finding details and the repeat count of every finding is preserved. Adding a new finding therefore does not change the weight of the existing ones. The operation is idempotent: even if the same approval is processed twice, the finding is added once.

The finding appears as its own line in the risk explanation, so it is readable why the band went up. Its severity is managed from the SQL Standards screen: the organization picks its own threshold, or deactivates the standard to switch the rule off entirely.

27.7 Password change on first login

The administrator account that ships with the installation must change its password at first login. The same rule applies to accounts whose password an administrator has reset.

The session token of a user who must change their password carries a separate claim. While that claim is present, a layer in the request pipeline rejects every request except change password and token refresh. That session can therefore do nothing but change the password.

The check is done on the server. The interface guiding the user to the password screen is a convenience, not a security boundary; even if the interface is bypassed the server rejects the request. The moment the password is changed the new token no longer carries the claim and the session returns to normal, with no need to sign in again.

Accounts whose password is still unchanged appear with their own badge in Settings > Users. That way an unattended account does not keep living with the installation password, and an administrator sees the gap simply by looking at the list.

27.8 Governance readiness screen

Settings > Diagnostics > Governance Readiness compares post-installation governance settings with the values expected in production. Every row is read live from its current value; the screen is not a stored report and shows the state of the moment it was opened. Its purpose is not to give a score but to make sure no setting silently stays at its default without anyone being aware of it.

The checks are grouped into nine areas: segregation of duties, rollback and sandbox, request discipline, identity and password, data protection, ITSM integration, audit and evidence, secret

placement, and items verified outside the product. The segregation of duties group counts staffing alongside settings: with strict settings but no second approver, the organisation locks itself out.

Every row also states where the setting is managed: the name of the screen and the exact key of the setting. The key is not translated, because it is the text you will search for. Rows whose right answer depends on the organization are marked Decide; those rows have no single correct answer, a conscious choice is enough. The screen imposes nothing, it shows whether a decision has been made.

The output can be shown to an auditor and kept as part of the installation acceptance record. The same list also exists in the deployment document; a list written into a document goes out of date over time, a list on screen does not.

The columns of a row

Each row has five columns. The check name says what is being asked; hovering over the name shows an explanation of why that check exists. The Managed in column gives the screen where the setting is changed and the exact key of the setting; the key is not translated, because it is the text you will search for on that screen. The Current value column shows the value in your installation. The Expected column shows the value the product expects. The Status column shows one of three states: In order, Decide or Attention.

What a dash in the Expected column means

A dash means there is no universally correct value for that row. For some settings the product does state a clear expectation: the expected answer to whether a requester may approve their own request is no, and the expected number of people who can approve is two or more. For other settings the right answer depends on the organisation, and the ticket number format is one of those. In one organisation numbers look like CR-12345, in another like INC0001234, and a third does not want a format check at all. On such rows the product does not impose a value; it only asks whether a deliberate decision has been made on the subject.

The three states and what you need to do

In order. The setting is suitable for production and there is nothing to do.

Decide. Nothing is wrong; what is missing is your decision. The Expected column shows a dash on these rows. The row moves to In order once you change the setting to fit your organisation, or once you deliberately confirm the current value by setting it yourself.

Attention. The setting is not at the value the product expects, or a control is not working. This is not a preference but something to be fixed. For example, if a regular expression that cannot be

compiled has been written, the format check is silently disabled; the row shows this as Attention rather than Decide, because the organisation believes the control is running.

The counters at the top of the screen

There are three counters above the list and they are mutually exclusive: a row is counted in exactly one of them. In order gives the rows that are at the expected value. Attention gives the rows the organisation needs to close. Awaiting a decision gives the rows where the organisation has to make its own choice or verify something outside the product.

The last two are kept apart for a reason: some of the rows awaiting a decision are rows the product can never close on its own. If they all landed in one counter, the Attention number would never reach zero and before long nobody would look at it.

Example: the ticket number format row

If this row shows Decide, the reason is that the value is still the pattern that shipped with the installation, which means it has not been reviewed against your own ticket format.

Where to look: the Settings > Parameters screen, key ChangeTicketRegex.

What to write: your organisation ticket number pattern as a regular expression. If your numbers look like INC0001234 you write `^INC\d{7}$`, and if they look like CR-12345 you write `^CR-\d{5}$`. It is worth putting a real example number in the sample format field next to it; that example is shown on screen when a user enters an invalid number.

What happens if you leave it empty: no format check is applied and any text in the ticket field is accepted. That is also a valid decision and moves the row to In order.

What happens if you do not change it: the check keeps working, but with the installation pattern. If your numbers do not match that pattern, users cannot enter their valid numbers and cannot save a request. This is why the row waits for a decision.

What happens if you write a wrong pattern: a pattern that cannot be compiled is not saved, and the error tells you where the problem is. If such a value is written directly into the database, the row shows Attention and the format check is not applied on that request.

The incident number format row (EmergencyTicketRegex) works the same way, with one difference: it ships empty. As there is no installation value to compare against, this row only warns when the pattern cannot be compiled. The product suggests no pattern here, because incident number formats differ between organisations and from the change ticket format.

Every row on the screen and what it means

The rows you may see on the screen are listed below in group order. Each entry gives the name shown on screen, the exact setting key in brackets, what the check asks, the expected value and where the setting is managed. Rows belonging to a feature you do not use may not appear at all; for example, if corporate sign-in is off and no provider is defined, the detail rows of that feature are not shown.

Segregation of duties

Can an approver execute (ApproverCanExecute). Whether the person who approved a request is allowed to execute that same request. The expected value is false: approving and executing should be done by different people. Managed in Settings > Parameters.

Can a requester approve their own request (RequesterCanSelfApproveExecute). Whether the person who raised a request may approve and execute it. The expected value is false. Managed in Settings > Parameters.

Distinct approver per step (RequireDistinctApproverPerStep). Whether one person can close more than one step in a multi step approval flow. The expected value is true: when off, a multi step flow can be completed by a single person and the assurance the extra steps were meant to give is lost. Managed in Settings > Parameters.

Users who can approve (cr.approve). The number of users holding the approval permission. The expected value is two or more. With strict segregation of duties settings but only one approver the organisation locks itself out: the requester cannot approve and there is nobody else. Managed in Settings > Users.

Users in the auditor role (Auditor). The number of users in the auditor role. The Expected column shows a dash: the auditor role is not mandatory and an organisation may run its audit through another role. The row moves to In order once at least one auditor exists. Managed in Settings > Users.

Rollback and sandbox

The value of these three rows is shown as how many servers have the flag on out of the total. If any server is marked as a production environment only production servers are counted; otherwise all active servers are. If the relevant module is not in your license the value is replaced by a note that the module is not licensed, because on an unlicensed module the server flag does nothing and showing it as on would be misleading. All three are managed on the server definition in Settings > Servers.

Rollback required (RollbackRequired). Whether a change request targeting this server can be executed without a complete rollback script. Expected: on for every server.

Automatic rollback generation (AutoRollbackEnabled). Whether the product generates the rollback script for object definition changes. Expected: on for every server.

Sandbox required (SandboxRequired). Whether a script must be tried on an isolated copy before it runs in production. Expected: on for every server.

Request discipline

Is a change ticket required (ChangeTicketRequired). Whether a change ticket number is asked for when a request is raised; it has three values: always, except on emergency requests, never. The Expected column shows a dash: the right answer depends on the organisation and all that is asked is that a conscious choice was made. Managed in Settings > Parameters.

Ticket number format (ChangeTicketRegex). The regular expression the ticket number has to match. If the example pattern that shipped with the installation is still in place, the row awaits a decision. If the pattern cannot be compiled the row shows Attention, because in that state the format check is silently disabled. Managed in Settings > Parameters.

Is an incident number required on emergency requests (EmergencyTicketRequired). Whether an incident number is asked for on an emergency change. The incident record is the evidence of why the normal path was shortened. The Expected column shows a dash. Managed in Settings > Parameters.

Incident number format (EmergencyTicketRegex). The pattern for the incident number; it ships empty and leaving it empty is a valid decision. This row only warns when the pattern you wrote cannot be compiled. Managed in Settings > Parameters.

ITSM integration group. Shows whether ITSM is used, whether the connection is defined and reachable, whether verification on save and on execution is on, whether the CAB approval is read separately, certificate validation, whether notes are being delivered and whether the password and the token are kept in environment variables. While ITSM is off, the main row shows Decide; not using ITSM can also be a deliberate decision. Details in Section 22.

Who may declare an emergency (EmergencyMinimumRole). The lowest role allowed to mark a request as an emergency. The expected value is a role higher than the requester role: declaring an emergency shortens the approval path and should not be available to everyone. Managed in Settings > Parameters.

Emergency approval policy (IsEmergencyPolicy). The number of approval policies marked as emergency policies. The expected value is one or more, but the count alone does not move the

row to In order: at least one emergency policy must contain a step that cannot be skipped. An emergency policy made only of skippable steps leads to unapproved change. Managed in Settings > Approval Policies.

Active servers with no approval policy (ServerApprovalPolicy). Counts the active servers that no approval policy covers and names the first five. The expected value is zero. A request cannot be saved for an uncovered server (rollback requests excepted), so no new work can be raised on that server. Managed in Settings > Approval Policies.

Critical object list (CriticalObjects). The number of objects marked as critical. The expected value is one or more. While the list is empty the critical object rules never fire: the rule engine appears to be working, but the most valuable tables in the organisation are unprotected. Managed in Settings > Critical Objects.

Identity and password

Minimum password length (Auth:Password:MinLength). The expected value is twelve or more. Managed in the configuration file.

Password history (Auth:Password:HistoryCount). How many previous passwords cannot be reused. The expected value is five or more; zero means a user can go straight back to the same password. Managed in the configuration file.

Uppercase requirement (Auth:Password:RequireUppercase). The expected value is true. When off, password complexity comes down to length alone. Managed in the configuration file.

MFA enrollment coverage (Mfa:Enabled). How many of the local and LDAP users with access have enrolled in multi factor authentication, shown as enrolled out of total. The expected value is everyone enrolled. This is a separate check for a reason: turning multi factor authentication on is not enough, the second factor is only asked of users who enrolled and a user who never enrolled signs in with a password alone. If multi factor authentication is off across the organisation the row is shown as a preference rather than a gap. Corporate accounts are not part of this rate: they cannot set up the product two-step verification, their multi-factor authentication is done by the identity provider and measured on the Corporate MFA is verified row. Tracked from Settings > Users.

MFA enrollment enforced (Mfa:EnforceEnrollment). Whether enrolling is mandatory. The expected value is true. When on, a user who has not enrolled can sign in but the session is good for the setup step alone and nothing else runs; the rule is enforced on the server. When off, enrolling is left to each user. Managed in the configuration file.

Setup administrator password changed (MustChangePassword). Whether the administrator account that shipped with the installation has changed its password. The expected value is true.

Until it does, the system that approves changes to production databases is protected by a password written in the installation document. Managed in Settings > Users.

Corporate sign-in (SSO) (Sso:Enabled). Whether corporate sign-in is on. The expected value is true, but the row is informational: corporate sign-in is not mandatory. When it is on, the access of a person who leaves is closed in one place at the organisation. Managed in the configuration file. While it is off and no identity provider is defined, the six rows below are not shown.

Corporate sign-in configuration errors (Sso:Providers). Lists the identity providers defined with errors and the field at fault. The expected value is zero; a provider with errors is not used. Managed in the configuration file.

Group to role mappings (SsoRoleMappings). The number of active mappings that belong to a configured identity provider; a mapping of a provider removed from the configuration is not counted, because it gives nobody a role. The expected value is one or more; without a mapping nobody arriving with a corporate account can get a role. Managed in Settings > Corporate Identity (SSO and LDAP).

Corporate MFA is verified (Sso:Providers:N:RequireMfaClaim). In how many identity providers the multi-factor verification statement is checked. The expected value is all providers. Managed in the configuration file.

Identity provider metadata is read over HTTPS (Sso:Providers:N:RequireHttpsMetadata). In how many identity providers the metadata must be read over HTTPS only. The product reads the signing keys of the identity provider from this address; if it is not HTTPS, someone in the middle can supply a fake key and sign in with a forged identity. The expected value is all providers. The value false is only for a local test. Managed in the configuration file.

Break-glass account (Auth:LocalLoginAllowedUsers). The number of local, password-protected accounts with the SystemManager role that can sign in when the identity provider is down. When the list is filled only the listed accounts count. The expected value is one or more. Managed in Settings > Users and in the configuration file.

Password sign-in limited to break-glass accounts (Auth:LocalLoginAllowedUsers). Shows the list of local accounts allowed on the password screen. The expected value is a defined list; an empty list means every local account can bypass the organisation's identity policy by signing in with a password. LDAP accounts also use the password screen and are not affected by this list; if live LDAP accounts exist their number is shown next to the value (for example admin | Ldap: 12) and the row warns. This is expected during a migration; when it is over, LDAP accounts are converted to corporate accounts. Managed in the configuration file.

The following directory (LDAP) rows appear only when the directory is in use (Ldap:Host set, Auth:Mode Ldap or at least one LDAP account). If the directory is configured but nobody uses it, the rows are for information.

Directory (LDAP) configuration errors (Ldap). Shows directory setting errors in the configuration file. The expected value is zero. Managed in the configuration file.

Directory connection is encrypted (Ldap:Security). The expected value is Ldaps or StartTls. With None, passwords travel over the network in clear text.

Directory certificate is validated (Ldap:AllowUntrustedCertificate). The expected value is false. true is only for a test environment with a self-signed certificate.

Role source of directory accounts (Ldap:RoleSource). For information: Manual or Directory.

Directory group to role mapping (SsoRoleMappings). The number of active mappings of the "Directory (LDAP)" provider. The expected value is one or more; it warns only when the role comes from the directory.

Groups are matched by full name (Ldap:GroupMatch). The expected value is Dn. With short-name matching, a group with the same name in another folder also grants the role.

Directory service account (Ldap:BindDn). The expected value is defined. Needed for synchronisation and the SearchDn format.

Directory synchronisation is on (Ldap:Sync:Enabled). The expected value is true. When off, the product account and the open session of a person disabled in the directory stay open.

Last directory synchronisation (Ldap:Sync:IntervalMinutes). The expected value is a synchronisation completed within twice the interval. If it stopped, nothing was changed; the reason is in the Ldap.SyncAborted record.

Product lockout threshold is below the directory (Auth:AccountLockoutMaxAttempts). The value is "product / directory"; the directory threshold is read at the last synchronisation. If the product does not lock before the directory does, password guessing through the product screen locks the person's directory account.

Data protection

Masking enabled (QrMaskingEnabled). Whether sensitive columns are masked on query result requests. The expected value is true. When off, neither detection nor masking is performed and only the record is kept. Managed in Settings > Parameters.

Result delivery mode (QrDeliveryMode). The organisation wide ceiling for the channel through which the result file and its password are delivered. The Expected column shows a dash,

because the right answer depends on the organisation. The row does warn, however, if the loosest mode is selected, that is if the package and its password travel in the same email: anyone who gets hold of that email can open the file. Managed in Settings > Parameters.

Servers exempt from masking (MaskingMode). Counts the active servers where masking by column name is disabled. The expected value is zero. That mode is meant for environments holding synthetic data; selected on a production server it means sensitive columns are not masked on the basis of their names. Managed in Settings > Servers.

Sensitive data patterns defined (SensitiveDataPatterns). The number of enabled sensitive data patterns. The expected value is one or more. With no patterns, content based detection has no rule to run. Managed in Settings > Sensitive Data Patterns.

Audit and evidence

Four eyes rule on settings changes (SettingsApprovalPolicy). Whether changes made on the settings screens go to a second person for approval. The expected value is either control settings only or all settings; switched off is not an expected value. Managed in Settings > Parameters.

Installation id (Evidence:Signing:InstallationId). This value is what proves the evidence dossier, the period package, the audit records package and the ledger digest came from the same installation. If left empty all of them are produced without an identity and the auditor comparison compares empty with empty and passes. Being non-empty is not enough either: if the development value shipped with the product is left unchanged the row asks for attention, because when two different organisations carry the same label, files standing in for one another cannot be told apart. It must be unique per installation, different per environment (for example ORG-PROD-01, ORG-TEST-01) and never changed afterwards; if it changes, older evidence files can no longer be verified together with newer ones.

Product version (Directory.Build.props). Shows the version written into the header of every evidence package. This row has no pass condition, it exists to make the value visible: support calls and auditor questions can then refer to the same value. The suffix identifies the exact code the build came from; it is the only thing that separates an installation that received a fix from one that did not under the same version number. The value comes from the product build and cannot be changed from a screen or a configuration file.

Evidence table is a ledger (Database:EvidenceLedger). Shows that audit records are kept in an append-only ledger table. Expected value: Append-only ledger table.

Evidence write boundary. Shows that there is no trigger on the evidence table, no change in the write procedure and no unexpected object in the evidence schema. If there is a finding, the finding codes are shown and change operations stop.

Last ledger digest (Evidence:Ledger:DigestIntervalMinutes). Shows when the last ledger digest was published. Needs attention if there has been no digest for longer than the digest interval multiplied by Evidence:Ledger:DigestStaleAfterIntervals.

Immutable store for digests (Evidence:Ledger:DigestOutputPath). If it is not defined, no record is shown as "covered by an external digest".

Where the signing key is kept (Evidence:Signing:Keys:{id}:CertificateThumbprint). The expected value is the Windows certificate store. Anyone who manages the server can copy a key from the configuration file or an environment variable.

This group also contains these rows: whether signing is enabled (Evidence:Signing:Enabled; with signing off, execution on a Production target is refused), the fingerprint of the signing key (Section 18.6), permission for unsigned Production execution (Evidence:Signing:AllowUnsignedProductionExecution; expected false), target server certificate validation, the governance state seal, time stamp health when time stamping is enabled, and nine SIEM rows (Section 25.8). The rollback and sandbox group also has a sandbox on production row (Sandbox:AllowOnProduction), and the items verified outside the product group has, when applicable, a row on the transport security of the worker endpoint.

Notification delivery (Mail:Enabled / Mail:DevMode). Whether notification emails are really being sent. The expected value is true. If development mode is on, every email is redirected to a single address and the row says so. When notifications do not go out, a request waiting for approval never reaches anyone: the flow stops and the reason is invisible. Managed in the configuration file.

Maintenance and archiving task (MaintenanceTasks). Whether at least one maintenance task is defined and enabled. The expected value is true. The row does not look at what the tasks do; there is no archiving or deletion task for the evidence ledger, and the ledger table does not accept deletes. Managed in Settings > Maintenance Tasks.

Secret placement (before production)

The row names in this group are not translations but the setting paths themselves; they are written as they are because they are the text you will search for. The value takes one of three states: configuration file, environment variable or not set. The expected value is the environment variable. The reason is this: the configuration file lives in source control, so a secret sitting there in plain text belongs to everyone who can reach the repository. A secret that is not mandatory and is not set at all is not counted as a gap, because that feature is simply not in use. The group is not a requirement but a list to be closed before going to production.

The list holds these secrets: the application database connection string, the token signing key, the server password encryption key, the query result package encryption key, the worker access key, the email password, the directory bind password and the result package password key. The client secret of every identity provider defined for corporate sign-in and, when a certificate is set, the SAML certificate password are listed as separate rows too. The first five are mandatory; the rest are only set if the related feature is in use.

Verified outside the product

The product cannot verify the rows in this group; they are set up in the network, the operating system and the infrastructure. The whole group is informational and does not affect the Attention counter. They are on the list so that an invisible risk is not left unconsidered during installation.

Worker endpoint restricted at network level (WorkerEndpointNetworkRestricted). This row is a declaration and there is no control running behind it. The worker configuration endpoint has to carry secrets so the worker can run; making it reachable only from the worker host is a firewall or reverse proxy rule and the product cannot see it. The value has three states: empty, yes and no. Empty says the decision was never made, no says the risk was knowingly accepted, and in an audit those two answers raise different questions. Saying yes changes nothing at the endpoint; what it does is write into the audit trail who made the decision and when. Because the declaration counts as a control setting, changing it goes to a second person for approval while the four eyes rule covers it. Managed in Settings > Parameters.

Worker endpoint access trace, last thirty days (Workers:ApiKey). This row is not a declaration but a measurement. Every request reaching the worker configuration endpoint, successful or not, is written to the audit trail together with its address. The row shows two numbers: how many distinct addresses called the endpoint and how many attempts used a wrong key. The expected value is one address and zero denied attempts. A second address says either the worker host moved or someone else is calling the endpoint. A denied attempt says the endpoint is visible from somewhere it does not belong; with the network restriction in place that request would never have reached it. The limit should be known as well: the address is read from the connection itself, not from a header that can be forged. So if a reverse proxy sits between the worker and the application, every call appears to come from the proxy and the distinct address count stays at one; the denied attempt count keeps working in that case.

DAM and SIEM matching declarations

In the group of items verified outside the product there are two advisory rows. Both record the decision of the organisation; the product cannot check the DAM configuration or whether the account is used somewhere else.

Row (parameter)	Options	When it passes
The SQL account of the product is dedicated (TargetAccountDedicated)	Yes, No	When Yes is selected
API and Worker addresses are in the DAM allowlist (DamAllowlistMaintained)	Yes, No, NotUsed	When Yes or NotUsed is selected (NotUsed: the organisation does not use DAM)

A row left empty shows "Decide". Both are advisory: when they do not pass, they do not raise the "Attention" counter at the top. Changing a declaration is a control change: if the four eyes rule is on it goes to approval, and who changed it and when is written to the audit trail and goes to the SIEM.

The description of the Delivery threshold and Streams sent rows carries this warning: if the threshold is above Notice, or the Audit stream is closed, the execution window events do not reach the SOC. This sentence is a fixed part of the description; it does not change with the value of the threshold.

27.9 Scope and boundary of the rule engine

The rule engine parses the script with the grammar of each target engine and works on sentence structure. It is therefore designed to catch what can be read from the script itself: an UPDATE without a filter, a DROP on a critical object, or a naming convention. A word inside a comment does not trigger a rule and a condition wrapped in parentheses is not missed. Limit: the PostgreSQL parser uses helper regular expressions for some checks; on Oracle, if grammar parsing fails the analysis falls back to a regular expression based method, which is less precise.

Whether a query uses an indexed column, however, cannot be read from the script; it requires looking at the target database catalogue and knowing the index definitions of the moment. That check is not in the scope of the rule engine today and belongs to database monitoring tools.

The boundary is deliberate. Producing warnings in an area where a certain answer cannot be given fills the screen with unreliable warnings, and over time people stop reading all of them. For that reason the rule engine does not produce warnings on matters that cannot be read from the script.

27.10 Disabled buttons and their reasons

If an action cannot be performed, the reason is written next to that button. The rule is this: hover over a disabled button and the tooltip states the reason as a full sentence. The user does not need a document, a log file or an administrator to find out why something is not possible.

Buttons behave in three ways. If you do not have the permission, the button is not shown at all: keeping an action on screen that you can never perform serves no purpose. If you have the

permission but the state of the request does not allow it, the button is shown disabled and the reason is in the tooltip. If the conditions are met, the button is enabled.

The reason in the tooltip is the same answer the server would give if you pressed the button. The screen and the command use the same rule; the reason on a disabled button is the real reason. An enabled button can still be rejected by checks made at the moment of the action (approval chain, ITSM, evidence write boundary); the message gives the reason.

The reason changes with the state of the request, and that difference matters. If the request is still live, the sentence tells you to wait: the change has not been applied yet for a review, the source request has not been executed yet for a rollback. If the request is closed, the sentence tells you there is nothing to do: this request was closed without being applied, there is no change to roll back, this request is closed and no further action can be taken. Writing the same sentence for every state would keep the user waiting for something that will never happen, so the reasons are separated by state. On a closed request there is also a one line information box at the top of the page; it states that the request is closed, in which state it closed and that no action can be taken, without hovering over anything. A user who wants to repeat the same work can copy the request.

The four eyes rule appears in the same way on settings screens. On a record with a change awaiting approval, editing and deleting are disabled; the tooltip of the Awaiting approval badge on that row states both the reason and a summary of the pending change.

27.11 Copying a request

Copying a request opens a new request. A copy is not more privileged than a request opened by hand: it passes through the same save gates. Whether the server is still defined and active, whether it is within the licence scope, whether the ticket rule of the organisation is satisfied, whether the script content complies with the rules, all of it is checked again.

The copy carries the scripts and the fields of the source. The approval flow is established afresh for the copy: the approval policy in force at that moment is applied and the starting state of the request follows that policy. The approval history, execution information and risk snapshot of the source are not carried over; the copy is a new event.

The emergency flag is not copied. Declaring an emergency is an authority and requires a matching emergency policy; a copy must not bypass those two controls. The justification and the incident number are not carried either: they belong to a specific incident, and the justification of a failure from months ago must not stick to a new request. The copy of an emergency request opens as a normal request and the screen states that the source was marked as an emergency and that this information was not carried over. If it is needed the user

sets the flag themselves, writes the current justification, and that path goes through both controls.

Rollback requests are not copied. A rollback always belongs to one specific request; a copy of it would turn into a pile of scripts with no clear answer to which change it reverses.

Section 28. Appendices

28.1 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)

In addition to the values in the table there is 8 = ReconciliationRequired. It is not a dead state but it is frozen: it is entered after an execution whose commit on the target is uncertain, it cannot be edited or executed again, and it is left only through a reconciliation decision. 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
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
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.

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

28.2 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	Always	Yes
ChangeTicketRegex	ITSM Ticket	^CR-\d{5}\$	Yes
EmergencyTicketRequired	ITSM Ticket	Never	Yes
EmergencyTicketRegex	ITSM Ticket	(empty)	Yes
QrFileRetentionDays	Query Result	1	Yes
QrRequesterCanDownload	Query Result	true	Yes
QrDownloadMinimumRole	Query Result	ChangeManager	Yes
QrDeliveryMode	Query Result	AttachmentPortalPassword	Yes
QrMaxRowCount	Query Result	1000000	Yes
QrOutputFormat	Query Result	Auto	No

Key	Group	Default	Control setting
QrCsvThresholdRowCount	Query Result	500000	No
QrExcelAutoFitMaxRows	Query Result	50000	No
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
SiemEnabled	SIEM	true	Yes
SiemStreams	SIEM	Audit;Anchor;AppLog;Heartbeat	Yes
SiemMinSeverity	SIEM	Notice	Yes
SiemIncludeScriptText	SIEM	false	Yes
RequestTicketRegex	ITSM Ticket	(empty)	Yes
ItsmEnabled	ITSM Ticket	false	Yes

Key	Group	Default	Control setting
ItsmVerifyOnSave	ITSM Ticket	Block	Yes
ItsmExecuteGate	ITSM Ticket	Block	Yes
ItsmEnforceWindow	ITSM Ticket	false	Yes
ItsmVerifyIncident	ITSM Ticket	Warn	Yes
ItsmQueryResultTicketKind	ITSM Ticket	Request	Yes
ItsmUnavailablePolicy	ITSM Ticket	AllowAndRecord	Yes
ItsmReadyStates	ITSM Ticket	Scheduled;Implement	Yes
ItsmClosedStates	ITSM Ticket	Review;Closed;Canceled;Cancelled;Resolved;Closed Complete;Closed Incomplete;Closed Skipped	Yes
ItsmApprovedValues	ITSM Ticket	Approved	Yes
ItsmWriteBackEnabled	ITSM Ticket	true	Yes
ItsmWriteBackEvents	ITSM Ticket	Request events	Yes

28.3 Permission key quick reference

One more key is not in the table: audit.anchorexport, which opens the export of the period package (Section 17). In the installation it is held by the Auditor and SystemManager roles.	Screen or action it enables
Key	
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

One more key is not in the table: audit.anchorexport, which opens the export of the period package (Section 17). In the installation it is held by the Auditor and SystemManager roles. Key	Screen or action it enables
audit.view	Audit log and verification actions
audit.evidenceexport	Exporting evidence dossiers
rp.manage	Release packages
settings.servers	Servers
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

28.4 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.

28.5 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.
- [] 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.
- [] Evidence signing has been configured (Evidence:Signing: key, InstallationId) and the public key fingerprint has been given to the auditor through a separate channel. With signing off, execution on a Production target is refused.

28.6 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.
- [] If ITSM is used, the connection has been tested, the state lists have been set together with the ITSM administrator and ItsmEnabled has been switched on.
- [] The segregation of duties parameters have been decided.
- [] A test request has been opened and taken end to end.

28.7 Audit preparation checklist

- [] The Control Exceptions report has been run and every row can be explained.
- [] If ITSM is used, the ITSM 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.
- [] Ledger Status is Healthy; the digest files of the period are in a store closed to deletion and change (such as WORM) and sys.sp_verify_database_ledger succeeded on a copy (Auditor Kit).
- [] Evidence dossiers have been produced and verified for sample requests.