



Relativity Data Grid Guide

October 30, 2024 | Version 24.0.315.10

For the most recent version of this document, visit our [documentation website](#).

Table of Contents

1 Relativity Data Grid	4
1.1 Getting started	4
1.2 Hardware and system requirements	6
1.3 Supported and unsupported functionality	6
1.3.1 Storage (as compared to SQL storage)	6
2 Installing Data Grid	7
2.1 Enabling your workspace and extracted text field for Data Grid	7
2.2 Data Grid agents	8
3 Backing up Relativity Data Grid	10
3.1 Backing up Elasticsearch	10
3.2 Creating a repository	10
3.3 Creating snapshots	12
3.3.1 Creating snapshots manually from within Sense	12
3.3.2 Scheduling a Windows task using Curator	13
3.4 Restoring a snapshot	29
3.4.1 Restoring snapshots from the Elasticsearch head console	29
3.4.2 Restoring snapshots with cURL	29
4 Data Grid Text Migration application	31
4.1 Special considerations	31
4.1.1 Before running text migration	31
4.1.2 Running text migration	32
4.1.3 Supported and unsupported functionality	32
4.2 Installing the Data Grid Text Migration application	33
4.2.1 Installing the text migration agents	33
4.3 Running text migration	33
4.4 Running text migration	33
4.4.1 Identifying workspaces to migrate	34
4.4.2 Identifying long text fields to migrate	34
4.4.3 Creating a text migration job	34
4.4.4 Running the Breakage Report	35

4.4.5 Creating and/or activating a dtSearch index	36
4.4.6 Resolving saved searches and views	37
4.4.7 Running a text migration job	38
4.4.8 Viewing migration errors	39

1 Relativity Data Grid

Relativity Data Grid allows you to store, search, and analyze extracted text and audit data at massive scale. Data Grid enables faster workflows through continuous indexing and distributed search to gain deeper insight to your extracted text and audit data. The benefits of using the Data Grid data store include:

- Scalability and more efficient review workflows on large case sizes and audits.
- More performant database maintenance, backup, and upgrades.
- A reduction in SQL server database sizes, leading to better performing SQL databases.
- Increased visibility into reviewer productivity and system performance.

Data Grid uses Elasticsearch to store and search audits.

Note: Text or audit information stored in Relativity Data Grid is inaccessible for some third-party applications. It's recommended that you contact any vendors of third-party applications to confirm their compatibility with Relativity Data Grid.

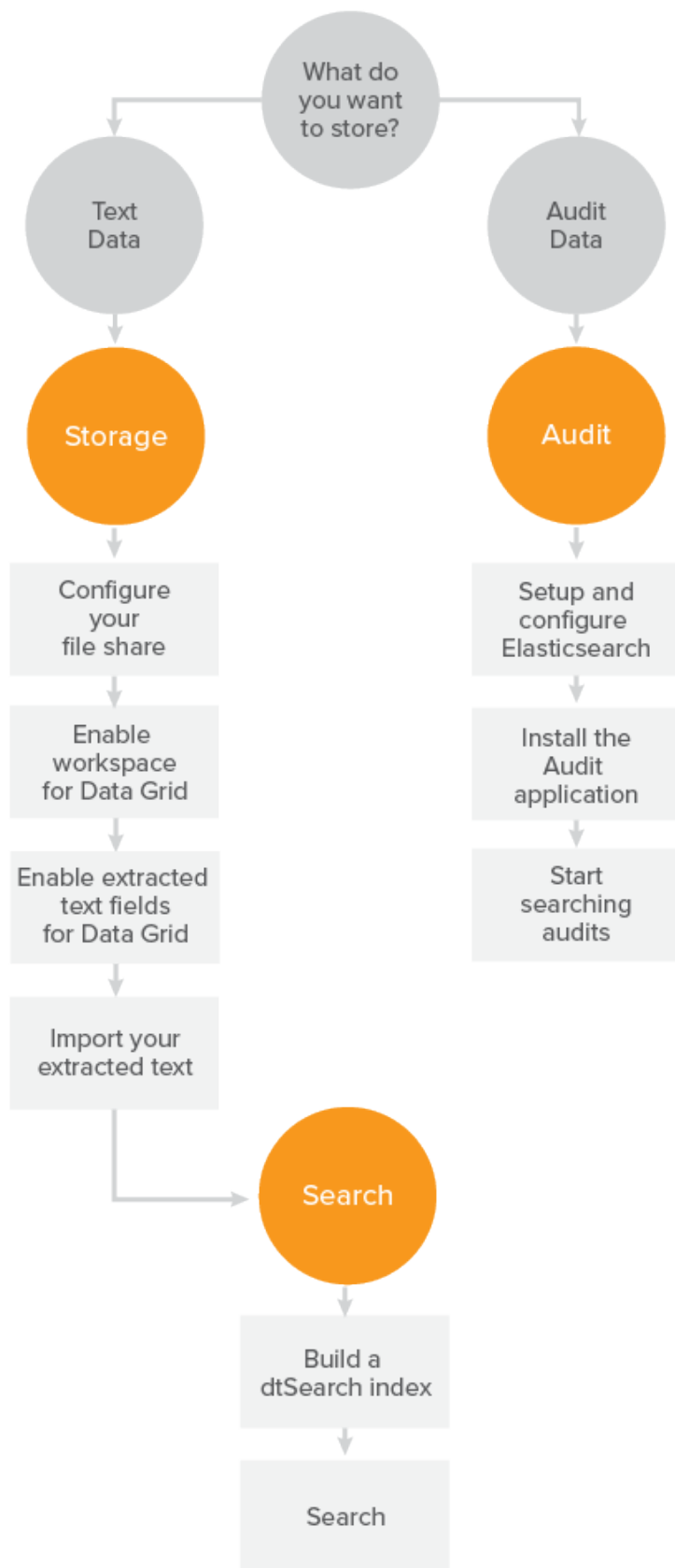
This page provides a brief description of Data Grid terminology, functionality, and important considerations to take into account before implementing Data Grid for new workspaces.

Note: Data Grid supports Windows servers only.

This page contains the following sections:

1.1 Getting started

Use the following workflow to get started with Data Grid.



1.2 Hardware and system requirements

The following table describes the hardware and system requirements per Data Grid feature.

	Storage	Audit
Hardware requirements	Fileshare	Elasticsearch
System requirements	Per Relativity specification	For more information about system requirements and infrastructure recommendations, see Elasticsearch system requirements.

1.3 Supported and unsupported functionality

The following sections describe supported and unsupported functionality for Data Grid.

1.3.1 Storage (as compared to SQL storage)

To store text in Data Grid, you can use the same default file share as your natives and images, or you can designate a file share specifically Data Grid text. Once you enable a long text field's access to Data Grid, you can't disable it, so it's important to understand the benefits and limitations of storing text in Data Grid.

Note: When you ARM restore to RelativityOne, extracted text is automatically stored in Data Grid.

Supported extracted text functionality	Currently unsupported functionality
<ul style="list-style-type: none">■ Import/export through the Relativity Desktop Client■ Viewer■ Preview■ OCR■ dtSearch indexing and searching■ Persistent highlight sets■ Processing■ Integration Points■ Analytics■ ARM	<ul style="list-style-type: none">■ Keyword search■ SQL queries to long text fields stored in Data Grid■ Adding extracted long fields stored in Data Grid to layouts (including the Document panel)■ RSAPI query■ Pivot and Sort in the UI■ Filtering in the Document list on extracted text■ Mass operations:<ul style="list-style-type: none">○ Edit○ Replace○ Tally/Sum/Average○ Export to File

2 Installing Data Grid

This page includes steps for configuring Data Grid Text (Core) in your environment. Before configuring Data Grid, ensure you've completed the pre-installation steps. For more information, see .

2.1 Enabling your workspace and extracted text field for Data Grid

To enable your workspace for Data Grid, perform the following steps:

Note: We recommend you only enable Data Grid for fields storing extracted text, OCR text, or translated text.

1. Navigate to the **Workspace Details** tab, and then click **Edit**.
2. Enable the **Is Data Grid Enabled** field.

The screenshot shows the 'Advanced Settings' tab for a workspace named 'Salt Vs Pepper'. Under the 'Resource Information' section, the 'Is Data Grid Enabled' toggle switch is turned on and highlighted with a red rectangular box. Other settings include 'Resource Pool' set to 'ResourcePool_T029B', 'Database Location' as 'ctus01412829W01.sql-y001.r1.kcura.com\ctus01412829W01', 'Default File Repository' as '\\files\T029B\Files\'', 'Data Grid File Repository' as an empty dropdown, 'Default Cache Location' as '\\files\T029B\Files\cache\'', and 'Download Handler URL' as 'Relativity.Distributed'.

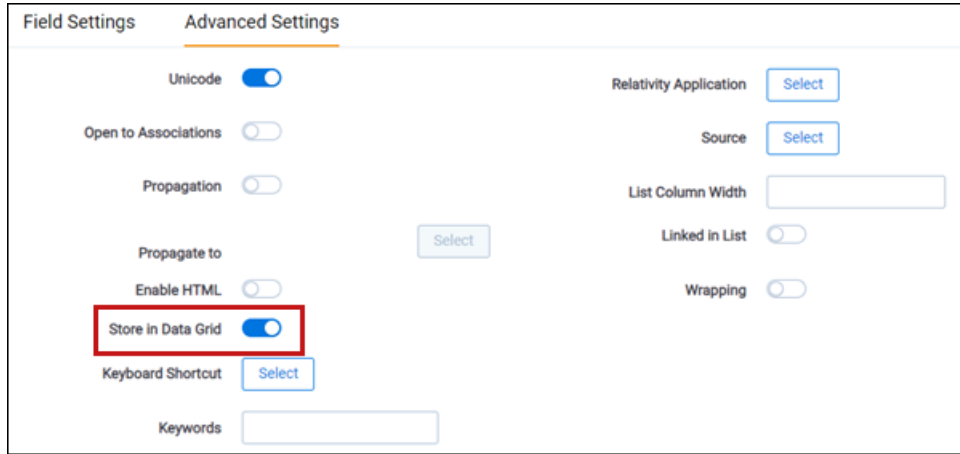
3. (Optional) Next to **Data Grid File Repository**, select the path for the physical location of the text files used by Data Grid. If no file repository is specified for this field, and Data Grid is enabled, Data Grid stores text in the default file repository.

Note: If you run out of space in this repository, you can specify a new repository. Data Grid will continue to read from the old repository as well as the new repository.

4. Click **Save**.

To enable the extracted text field for Data Grid, perform the following steps:

1. Navigate to the **Fields** tab.
2. Locate the extracted text field and click the **Edit** link next to it.
3. Enable the **Store in Data Grid** field under the Advanced Settings tab.



Note: If you are storing extracted text in Data Grid, the **Include in Text Index** field is set to No because there is no SQL text index. If you want to search using dtSearch, you must follow best practice of creating a saved search of fields you want to index.

4. Click **Save**.

Note: Enabling extracted text fields for Data Grid works for new workspaces only. You can't enable Data Grid for fields that already have text in SQL. If you want to migrate fields from SQL to Data Grid, you must use the Data Grid Text Migration application.

2.2 Data Grid agents

A number of agents are available to facilitate Data Grid operations in your environment. Ensure that the following agents are installed in your environment. For more information on installing agents, see the Agents Guide.

Agent name	Requirement information	Function	Agent type
Data Grid Manager	Only 1 per environment	A Data Grid Manager agent is an off-hours agent responsible for Data Grid enabled workspace management, including deleting outdated search results cache tables and monitoring Data Grid index conditions.	Single-installation

Agent name	Requirement information	Function	Agent type
Data Grid Worker	At least 1 per environment.	A Data Grid worker agent is part of building the Data Grid File Repository.	Multiple-installation

3 Backing up Relativity Data Grid

Data Grid stores text data in your file share. There is no automatic redundant storage of document text. Because Data Grid text data is stored in your file share, you may need to adjust your file system data backup frequency to meet the requirements of your Service Level Agreements (SLAs) and disaster recovery plans.

3.1 Backing up Elasticsearch

We recommend routinely backing up your data. Elasticsearch replicas provide high availability during run time, allowing toleration of sporadic node loss without interruption of service, but replicas don't provide protection against catastrophic failures. Create a complete backup of the entire cluster to protect your data if something goes wrong.

You can use the snapshot API to create a backup of the cluster. The snapshot API saves the current state of all data in your cluster to a shared repository. The first snapshot you create is a complete copy of all data on the cluster.

Each subsequent snapshot compares the current state of the data in the cluster to the data stored in the repository and only modifies the differences between the two.

The snapshot API incrementally edits the repository each time you create a new snapshot, so subsequent backups are significantly faster since they require less data transmission. This page explains all steps necessary to back up and restore Elasticsearch.

Note: Data Grid supports Windows servers only.

3.2 Creating a repository

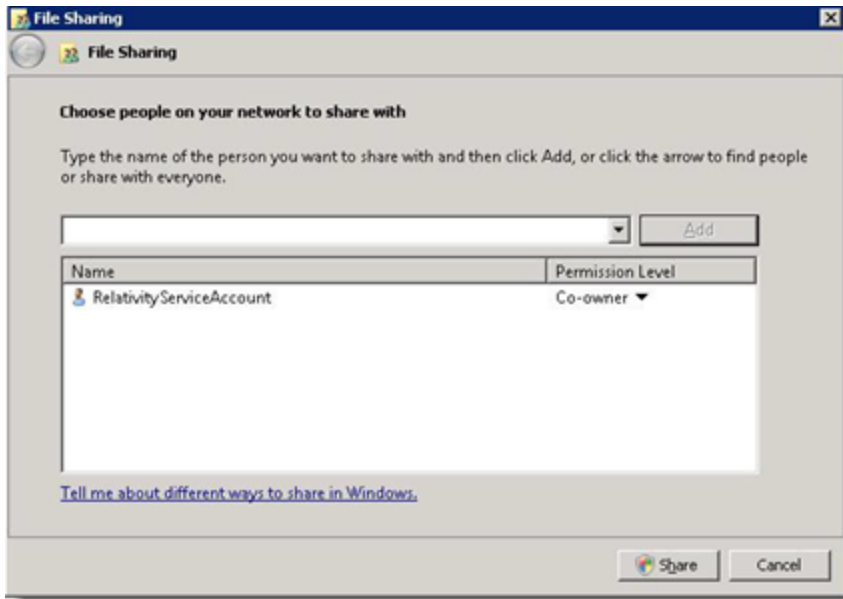
Before implementing this backup method, you must create a repository that can store snapshots. You can use any of the following four repository types:

- Shared file system, such as a NAS
- Amazon S3
- Hadoop Distributed File System (HDFS)
- Azure/Entra Cloud

Use the following steps to create and share a folder:

1. Create a folder named **ElasticBackup** to store snapshots. (//COMPUTER_NAME.business.corp/ElasticBackup).
2. Right-click on the folder, and then click **Properties**.
3. Select the **Sharing** tab, and then click **Share**.

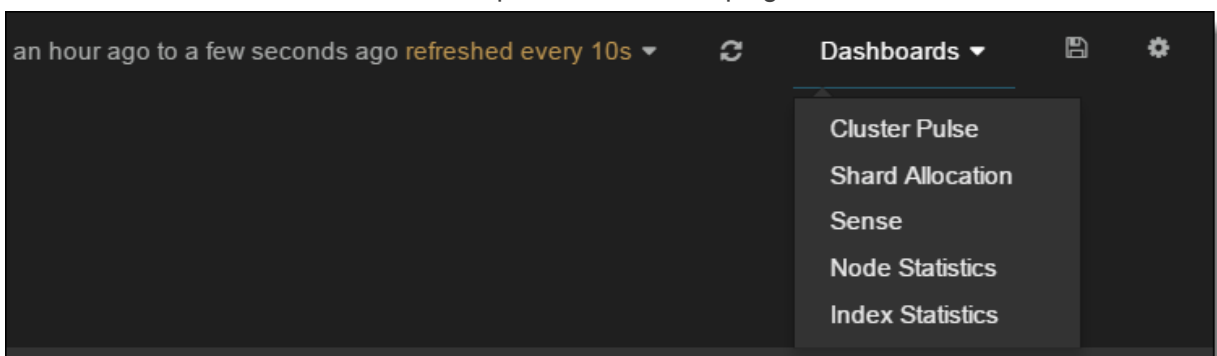
4. Enter the user that runs the Elasticsearch Windows service (domain\account), and then click **Add**.



5. Select the user on the share list and set the **Permission Level** to **Co-owner**.
6. Click **Share**.
7. When the share completes, click **Done**.
8. On the **Document Properties** dialog, select the **Security** tab.
9. Verify that the user that runs the Elasticsearch Windows service has **Full Control** security permissions to the folder.

Use the following steps to link Elasticsearch to the repository folder:

1. Launch **Marvel** from within a browser to connect to one of the nodes in your cluster.
2. Launch **Sense** from the Dashboards drop-down near the top right.



3. Edit the location value and run the following to set up a shared file system repository:

```
PUT /_snapshot/my_backup
{
  "type": "fs",
```

```
"settings": {
  "location": "//COMPUTER_NAME.business.corp/Shared/ElasticBackup",
  "compress": true
}
```

4. Verify your snapshot settings exist by performing the following call:

```
GET /_snapshot/
```

3.3 Creating snapshots

There are two ways to create snapshots:

- Creating snapshots manually from within Sense
- Scheduling a Windows task using Curator

3.3.1 Creating snapshots manually from within Sense

Run the following to back up all open indexes into a snapshot named "snapshot_1".

```
PUT /_snapshot/ElasticBackup/snapshot_1
```

Note: Increment the name of the snapshot for best results (e.g., snapshot_1, snapshot_2, snapshot_3, etc.). All alphabetical characters in the snapshot name must be lowercase.

Verify that this process created a backup by navigating to the following location:

```
//COMPUTER_NAME.business.corp/Shared/ElasticBackup
```

Your backup should look similar to the following image:

AuditUI ▶ ElasticBackup ▶ sharktopus_edds1130643_audit ▶ 2

urn New folder

Name	Date modified	Type	Size
_0	7/3/2014 2:16 PM	File	1 KB
_1	7/3/2014 2:16 PM	File	1 KB
_2	7/3/2014 2:16 PM	File	2 KB
_3	7/3/2014 2:16 PM	File	1 KB
_4	7/3/2014 2:16 PM	File	1 KB
_5	7/3/2014 2:16 PM	File	31 KB
_6	7/3/2014 2:16 PM	File	509 KB
_7	7/3/2014 2:16 PM	File	8 KB
_8	7/3/2014 2:16 PM	File	1 KB
_9	7/3/2014 2:16 PM	File	120 KB
_a	7/3/2014 2:16 PM	File	46 KB
_b	7/3/2014 2:16 PM	File	1 KB
_c	7/3/2014 2:16 PM	File	46 KB
_d	7/3/2014 2:16 PM	File	1 KB
_e	7/3/2014 2:16 PM	File	1 KB
_f	7/3/2014 2:16 PM	File	70 KB
_g	7/3/2014 2:16 PM	File	1 KB
_h	7/3/2014 2:16 PM	File	1 KB
_i	7/3/2014 2:16 PM	File	4 KB
_j	7/3/2014 2:16 PM	File	1 KB
_k	7/3/2014 2:16 PM	File	1 KB
_l	7/3/2014 2:16 PM	File	1 KB
_m	7/3/2014 2:16 PM	File	470 KB
_n	7/3/2014 2:16 PM	File	2 KB
_o	7/3/2014 2:16 PM	File	1 KB
_p	7/3/2014 2:16 PM	File	1 KB
_q	7/3/2014 2:16 PM	File	3 KB
_r	7/3/2014 2:16 PM	File	1 KB
_s	7/3/2014 2:16 PM	File	216 KB
_t	7/3/2014 2:16 PM	File	16 KB
_u	7/3/2014 2:16 PM	File	65 KB
_v	7/3/2014 2:16 PM	File	93 KB
_w	7/3/2014 2:16 PM	File	597 KB
_x	7/3/2014 2:16 PM	File	1 KB
snapshot-snapshot_1	7/3/2014 2:16 PM	File	4 KB

For more information on snapshot commands, including the ability to snapshot specific indexes, see [Backing up your cluster](#) on the Elasticsearch website for your version of Elasticsearch.

3.3.2 Scheduling a Windows task using Curator

The best way to schedule automatic backups of your data is to use Curator, which you can combine with scheduled tasks to automatically invoke the desired behavior.

The Curator Python API can be used to manage indexes and snapshots with the following features:

- **Iterative methods** - allow you to retrieve data across the cluster within specified parameters.
- **Non-iterative methods** - allow you to retrieve data within a single index or snapshot.
- **Helper methods** - allow you to retrieve values required to complete iterative and non-iterative methods.

For more information on Curator and snapshot capabilities, see [Snapshot](#) on Github.

3.3.2.1 Installing Curator 4

Before setting up Curator, you must complete the following:

1. Download and run the [Curator installer from Elastic](#).
2. Download and install the [Microsoft C++ redistributable](#).

3.3.2.2 Running Curator

Once you install Curator, you can use it to run “actions” which are created in the action.yml file. Use the following command to run an operation in the action.yml file:

Note: This command runs a dry-run where Curator simulates the action(s) in the file without making any changes. To actually run the operation, remove the **--dry-run** flag.

```
curator [--config CONFIG.YML] [--dry-run] ACTION_FILE.YML
```

This command references both a configuration and action file. You can run PowerShell scripts to automatically create the files needed to run Curator.

Running Curator from PowerShell

The following PowerShell scripts automatically create the PS1 and YML files needed to run Curator. It also sets up three Windows tasks: backup, backup cleanup, and Marvel cleanup.

Click to expand the Curator setup without SSL

Run the following in Powershell:

```
<##>
Param(
    $probHostName = "masternodename",
    $marvelHostName = "marvelnodename",
    $repositoryName = "datagridbackup",
    [string]$esUsername = "esadmin",
    [string]$esPassword = "esadmin",
    [string]$marvelUsername = "marvel",
    [string]$marvelPassword = "marvel",
    $curatorMsiPath = "C:\Curator\elasticsearch-curator-4.2.6-win32.msi",
    $curatorPath = "C:\Curator",
    $useSSL = "False"
)

$httpAuth = [system.Text.Encoding]::ASCII
$httpAuth = "$esUsername`:$esPassword"
$httpAuthMarvel = [system.Text.Encoding]::ASCII
$httpAuthMarvel = "$marvelUsername`:$marvelPassword"
$powerShellPath = (Resolve-Path /Windows/System32/WindowsPowerShell/v1.0/).Path

function IsCuratorInstalled {
    $isInstalled = ((Get-ChildItem "HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall") | Where-Object
```

```

{ $_.GetValue( "DisplayName" ) -eq "elasticsearch-curator" } )

    If (!$isInstalled) {
        Try{
            Write-Host "Installing Curator 4.2.6" -ForegroundColor Green
            Start-Process msixec.exe -Wait -ArgumentList "/I $curatorMsiPath /quiet" -ErrorAction Stop
            Write-Host "Installation Complete."
        }Catch{
            $ErrorMessage = $_.Exception.Message; Break
        }
    }else{
        Write-Host "Curator is already installed." -ForegroundColor Green
    }
}
IsCuratorInstalled

function MakeFiles
{
function MakeProdConfigYML
{
$prodConfigYml = @"
---
# Remember, leave a key empty if there is no value.  None will be a string,
# not a Python "NoneType"
client:
  hosts:
  - $probHostName
  port: 9200
  url_prefix:
  use_ssl: $useSSL
  certificate:
  client_cert:
  client_key:
  ssl_no_validate: True
  http_auth: $httpAuth
  timeout: 30
  master_only: False

logging:
  loglevel: INFO
  logfile:
  logformat: default
  blacklist: ['elasticsearch', 'urllib3']
"@

$prodConfigYml | Out-File .\prodConfig.yml -Encoding ascii
}
MakeProdConfigYML

function MakeBackupActionYML
{
$prodBackupYml = @"
# Remember, leave a key empty if there is no value.  None will be a string,
# not a Python "NoneType"
#
# Also remember that all examples have 'disable_action' set to True.  If you
# want to use this action as a template, be sure to set this to False after
# copying it.
# Leaving name blank will result in the default 'curator-%Ym%d%H%M%S'
actions:

```

```

1:
  action: snapshot
  description: "Created snapshots for all indexes."
  options:
    repository: $repositoryName
    name:
    ignore_unavailable: False
    include_global_state: True
    partial: False
    wait_for_completion: True
    skip_repo_fs_check: False
    timeout_override:
    continue_if_exception: False
    disable_action: False
  filters:
  - filtertype: none
"@

$prodBackupYml | Out-File .\prodBackup.yml -Encoding ascii
}
MakeBackupActionYML

function MakeCleanUpActionYML
{
$prodCleanUpYml = @"
# Remember, leave a key empty if there is no value.  None will be a string,
# not a Python "NoneType"
#
# Also remember that all examples have 'disable_action' set to True.  If you
# want to use this action as a template, be sure to set this to False after
# copying it.
# Leaving name blank will result in the default 'curator-%Y%m%d%H%M%S'
actions:
  1:
    action: delete_snapshots
    description: "Deletes snapshots older than 14 days."
    options:
      repository: $repositoryName
      retry_interval:
      retry_count:
      timeout_override:
      continue_if_exception: False
      disable_action: False
    filters:
    - filtertype: pattern
      kind: prefix
      value: curator-
      exclude:
    - filtertype: age
      source: creation_date
      direction: older
      unit: days
      unit_count: 14
      exclude:
"@

$prodCleanUpYml | Out-File .\prodCleanUp.yml -Encoding ascii
}
MakeCleanUpActionYML

```



```

function MakeMarvelConfigYML
{
$marvelConfigYml = @"
---
# Remember, leave a key empty if there is no value.  None will be a string,
# not a Python "NoneType"
client:
  hosts:
  - $marvelHostName
  port: 9200
  url_prefix:
  use_ssl: False
  certificate:
  client_cert:
  client_key:
  ssl_no_validate: False
  http_auth: $httpAuthMarvel
  timeout: 30
  master_only: False

logging:
  loglevel: INFO
  logfile:
  logformat: default
  blacklist: ['elasticsearch', 'urllib3']
"@

$marvelConfigYml | Out-File .\marvelConfig.yml -Encoding ascii
}
MakeMarvelConfigYML

function MakeMarvelCleanUpYML
{
$marvelCleanUpYml = @"
# Remember, leave a key empty if there is no value.  None will be a string,
# not a Python "NoneType"
#
# Also remember that all examples have 'disable_action' set to True.  If you
# want to use this action as a template, be sure to set this to False after
# copying it.
# Leaving name blank will result in the default 'curator-%Y%m%d%H%M%S'
actions:
  1:
    action: delete_indices
    description: "Deletes old marvel indexes"
    options:
      timeout_override:
      continue_if_exception: False
      disable_action: False
    filters:
    - filtertype: pattern
      kind: regex
      value: .marvel-es-
      exclude:
    - filtertype: age
      source: creation_date
      direction: older
      unit: days
      unit_count: 30
      exclude:

```

```

"@

$marvelCleanupYml | Out-File .\marvelCleanup.yml -Encoding ascii
}
MakeMarvelCleanupYML

function MakeBackupActions
{
$prodBackup = @"
cd\
cd ".\Program Files\elasticsearch-curator"
.\curator --config $curatorPath\prodConfig.yml $curatorPath\prodBackup.yml
"@

$prodBackup | Out-File .\prodBackup.ps1
}
MakeBackupActions

function MakeCleanupActionProd
{
$prodCleanupPs = @"
cd\
cd ".\Program Files\elasticsearch-curator"
.\curator --config $curatorPath\prodConfig.yml $curatorPath\prodCleanup.yml
"@

$prodCleanupPs | Out-File .\prodCleanup.ps1
}
MakeCleanupActionProd

function MakeCleanupActionMarvel
{
$marvelCleanup = @"
cd\
cd ".\Program Files\elasticsearch-curator"
.\curator --config $curatorPath\marvelConfig.yml $curatorPath\marvelCleanup.yml
"@

$marvelCleanup | Out-File .\marvelCleanup.ps1
}
MakeCleanupActionMarvel
}
MakeFiles

<#
    Make scheduled tasks.
#>

function MakeScheduledTaskDataGridBackup
{
$action = New-ScheduledTaskAction -Execute ("powershellPath" + "powershell.exe") -WorkingDirectory $curatorPath -Argument ".\prodBackup.ps1"
$trigger = New-JobTrigger -Once -At (Get-Date).date -RepeatIndefinitely -RepetitionInterval (New-TimeSpan -Hours 6)
$settings = New-ScheduledTaskSettingsSet -AllowStartIfOnBatteries -DontStopIfGoingOnBatteries -WakeToRun
Register-ScheduledTask -Action $action -Trigger $trigger -RunLevel Highest -TaskName "Data Grid Backup" -Description "Scheduled snapshot of the Production Data Grid cluster." -Settings $settings
}

MakeScheduledTaskDataGridBackup

```

```

function MakeScheduledTaskDataGridBackupCleanup
{
$action = New-ScheduledTaskAction -Execute ("$powerShellPath" + "powershell.exe") -WorkingDirectory $curatorPath -Argument ".\prodCleanup.ps1"
$trigger = New-JobTrigger -Once -At (Get-Date).date -RepeatIndefinitely -RepetitionInterval (New-TimeSpan -Hours 24)
$settings = New-ScheduledTaskSettingsSet -AllowStartIfOnBatteries -DontStopIfGoingOnBatteries -WakeToRun
Register-ScheduledTask -Action $action -Trigger $trigger -RunLevel Highest -TaskName "Data Grid Backup Cleanup" -Description "Scheduled clean up job for the Production Data Grid cluster."
}

MakeScheduledTaskDataGridBackupCleanup

function MakeScheduledTaskMarvelCleanup
{
$action = New-ScheduledTaskAction -Execute ("$powerShellPath" + "powershell.exe") -WorkingDirectory $curatorPath -Argument ".\marvelCleanup.ps1"
$trigger = New-JobTrigger -Once -At (Get-Date).date -RepeatIndefinitely -RepetitionInterval (New-TimeSpan -Hours 24)
$settings = New-ScheduledTaskSettingsSet -AllowStartIfOnBatteries -DontStopIfGoingOnBatteries -WakeToRun
Register-ScheduledTask -Action $action -Trigger $trigger -TaskName "Marvel Cleanup" -Description "Deletes indexes on the marvel cluster older than 30 days."
}

MakeScheduledTaskMarvelCleanup

```

Click to expand the Curator setup with SSL
Run the following in Powershell:

```

<##>
Param(
$probHostName = "dg-ramp-01",
$marvelHostName = "dg-ramp-agt",
$repositoryName = "datagridbackup",
[string]$esUsername = "esadmin",
[string]$esPassword = "esadmin",
[string]$marvelUsername = "marvel",
[string]$marvelPassword = "marvel",
$curatorMsiPath = "C:\Curator\elasticsearch-curator-4.2.6-win32.msi",
$curatorPath = "C:\Curator",
$useSSL = "True"
)

$httpAuth = [system.Text.Encoding]::ASCII
$httpAuth = "$esUsername`:$esPassword"
$httpAuthMarvel = [system.Text.Encoding]::ASCII
$httpAuthMarvel = "$marvelUsername`:$marvelPassword"
$powerShellPath = (Resolve-Path /Windows/System32/WindowsPowerShell/v1.0/).Path

function IsCuratorInstalled {
    $isInstalled = ((Get-ChildItem "HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall") | Where-Object { $_.GetValue( "DisplayName" ) -eq "elasticsearch-curator" } )

    If (!$isInstalled) {
        Try{
            Write-Host "Installing Curator 4.2.6" -ForegroundColor Green
            Start-Process msiexec.exe -Wait -ArgumentList "/I $curatorMsiPath /quiet" -ErrorAction Stop
            Write_host "Installation Complete."
        }
    }
}

```

```

    }Catch{
        $ErrorMessage = $_.Exception.Message; Break
    }
}else{
    Write-Host "Curator is already installed." -ForegroundColor Green
}
}
IsCuratorInstalled

function MakeFiles
{
function MakeProdConfigYML
{
$prodConfigYml = @"
---
# Remember, leave a key empty if there is no value.  None will be a string,
# not a Python "NoneType"
client:
  hosts:
    - $probHostName
  port: 9200
  url_prefix:
  use_ssl: $useSSL
  certificate:
  client_cert:
  client_key:
  ssl_no_validate: True
  http_auth: $httpAuth
  timeout: 30
  master_only: False

logging:
  loglevel: INFO
  logfile:
  logformat: default
  blacklist: ['elasticsearch', 'urllib3']
"@

$prodConfigYml | Out-File .\prodConfig.yml -Encoding ascii
}
MakeProdConfigYML

function MakeBackupActionYML
{
$prodBackupYml = @"
# Remember, leave a key empty if there is no value.  None will be a string,
# not a Python "NoneType"
#
# Also remember that all examples have 'disable_action' set to True.  If you
# want to use this action as a template, be sure to set this to False after
# copying it.
# Leaving name blank will result in the default 'curator-%Y%m%d%H%M%S'
actions:
  1:
    action: snapshot
    description: "Created snapshots for all indexes."
    options:
      repository: $repositoryName
      name:
      ignore_unavailable: False

```

```

    include_global_state: True
    partial: False
    wait_for_completion: True
    skip_repo_fs_check: False
    timeout_override:
      continue_if_exception: False
      disable_action: False
  filters:
  - filtertype: none
"@

$prodBackupYml | Out-File .\prodBackup.yml -Encoding ascii
}
MakeBackupActionYML

function MakeCleanUpActionYML
{
$prodCleanUpYml = @"
# Remember, leave a key empty if there is no value.  None will be a string,
# not a Python "NoneType"
#
# Also remember that all examples have 'disable_action' set to True.  If you
# want to use this action as a template, be sure to set this to False after
# copying it.
# Leaving name blank will result in the default 'curator-%Y%m%d%H%M%S'
actions:
  1:
    action: delete_snapshots
    description: "Deletes snapshots older than 14 days."
    options:
      repository: $repositoryName
      retry_interval:
      retry_count:
      timeout_override:
      continue_if_exception: False
      disable_action: False
    filters:
    - filtertype: pattern
      kind: prefix
      value: curator-
      exclude:
    - filtertype: age
      source: creation_date
      direction: older
      unit: days
      unit_count: 14
      exclude:
"@

$prodCleanUpYml | Out-File .\prodCleanUp.yml -Encoding ascii
}
MakeCleanUpActionYML

function MakeMarvelConfigYML
{
$marvelConfigYml = @"
---
# Remember, leave a key empty if there is no value.  None will be a string,
# not a Python "NoneType"
client:

```

```

hosts:
- $marvelHostName
port: 9200
url_prefix:
use_ssl: False
certificate:
client_cert:
client_key:
ssl_no_validate: False
http_auth: $httpAuthMarvel
timeout: 30
master_only: False

logging:
  loglevel: INFO
  logfile:
  logformat: default
  blacklist: ['elasticsearch', 'urllib3']
"@

$marvelConfigYml | Out-File .\marvelConfig.yml -Encoding ascii
}
MakeMarvelConfigYML

function MakeMarvelCleanUpYML
{
$marvelCleanUpYml = @"
# Remember, leave a key empty if there is no value.  None will be a string,
# not a Python "NoneType"
#
# Also remember that all examples have 'disable_action' set to True.  If you
# want to use this action as a template, be sure to set this to False after
# copying it.
# Leaving name blank will result in the default 'curator-%Y%m%d%H%M%S'
actions:
  1:
    action: delete_indices
    description: "Deletes old marvel indexes"
    options:
      timeout_override:
      continue_if_exception: False
      disable_action: False
    filters:
    - filtertype: pattern
      kind: regex
      value: .marvel-es-
      exclude:
    - filtertype: age
      source: creation_date
      direction: older
      unit: days
      unit_count: 30
      exclude:
"@

$marvelCleanUpYml | Out-File .\marvelCleanUp.yml -Encoding ascii
}
MakeMarvelCleanUpYML

function MakeBackupActions

```

```

{
$prodBackUp = @"
cd\
cd ".\Program Files\elasticsearch-curator"
.\curator --config $curatorPath\prodConfig.yml $curatorPath\prodBackup.yml
"@

$prodBackUp | Out-File .\prodBackup.ps1
}
MakeBackupActions

function MakeCleanUpActionProd
{
$prodCleanUpPs = @"
cd\
cd ".\Program Files\elasticsearch-curator"
.\curator --config $curatorPath\prodConfig.yml $curatorPath\prodCleanUp.yml
"@

$prodCleanUpPs | Out-File .\prodCleanup.ps1
}
MakeCleanUpActionProd

function MakeCleanUpActionMarvel
{
$marvelCleanUp = @"
cd\
cd ".\Program Files\elasticsearch-curator"
.\curator --config $curatorPath\marvelConfig.yml $curatorPath\marvelCleanUp.yml
"@

$marvelCleanUp | Out-File .\marvelCleanup.ps1
}
MakeCleanUpActionMarvel
}
MakeFiles

<#
    Make scheduled tasks.
#>

function MakeScheduledTaskDataGridBackup
{
$action = New-ScheduledTaskAction -Execute ("$powerShellPath" + "powershell.exe") -WorkingDirectory $cur-
atorPath -Argument ".\prodBackup.ps1"
$trigger = New-JobTrigger -Once -At (Get-Date).date -RepeatIndefinitely -RepetitionInterval (New-TimeSpan -
Hours 6)
$settings = New-ScheduledTaskSettingsSet -AllowStartIfOnBatteries -DontStopIfGoingOnBatteries -WakeToRun
Register-ScheduledTask -Action $action -Trigger $trigger -RunLevel Highest -TaskName "Data Grid Backup" -
Description "Scheduled snapshot of the Production Data Grid cluster." -Settings $settings
}

MakeScheduledTaskDataGridBackup

function MakeScheduledTaskDataGridBackupCleanup
{
$action = New-ScheduledTaskAction -Execute ("$powerShellPath" + "powershell.exe") -WorkingDirectory $cur-
atorPath -Argument ".\prodCleanUp.ps1"
$trigger = New-JobTrigger -Once -At (Get-Date).date -RepeatIndefinitely -RepetitionInterval (New-TimeSpan -

```

```

Hours 24)
$settings = New-ScheduledTaskSettingsSet -AllowStartIfOnBatteries -DontStopIfGoingOnBatteries -WakeToRun
Register-ScheduledTask -Action $action -Trigger $trigger -RunLevel Highest -TaskName "Data Grid Backup
Cleanup" -Description "Scheduled clean up job for the Production Data Grid cluster."
}

MakeScheduledTaskDataGridBackupCleanup

function MakeScheduledTaskMarvelCleanup
{
$action = New-ScheduledTaskAction -Execute ("powershell.exe") -WorkingDirectory $cur-
atorPath -Argument ".\marvelCleanUp.ps1"
$trigger = New-JobTrigger -Once -At (Get-Date).date -RepeatIndefinitely -RepetitionInterval (New-TimeSpan -
Hours 24)
$settings = New-ScheduledTaskSettingsSet -AllowStartIfOnBatteries -DontStopIfGoingOnBatteries -WakeToRun
Register-ScheduledTask -Action $action -Trigger $trigger -TaskName "Marvel Cleanup" -Description "Deletes
indexes on the marvel cluster older than 30 days."
}

MakeScheduledTaskMarvelCleanup

```

Running Curator manually

The following sections contain examples of the different action files you can create manually to run in Curator. You can save these YAML files anywhere. Ensure you use a full path to the files when executing the command.

Sample configuration file

Click to expand a sample config.yml file containing one master node.

```

---
# Remember, leave a key empty if there is no value. None will be a string,
# not a Python "NoneType"
client:
  hosts:
    - cs-mv1-agtfs
  port: 9200
  url_prefix:
  use_ssl: False
  certificate:
  client_cert:
  client_key:
  ssl_no_validate: False
  http_auth:
  timeout: 30
  master_only: False

logging:
  loglevel: INFO
  logfile:
  logformat: default
  blacklist: ['elasticsearch', 'urllib3']

```

Sample backup action file

Click to expand a sample action_file.yml file that runs a backup action.

```

# Remember, leave a key empty if there is no value. None will be a string,
# not a Python "NoneType"
#

```



```

# Also remember that all examples have 'disable_action' set to True. If you
# want to use this action as a template, be sure to set this to False after
# copying it.
# Leaving name blank will result in the default 'curator-%Y%m%d%H%M%S'
actions:
  1:
    action: snapshot
    description: "description"
    options:
      repository: datagridbackup
      name:
      ignore_unavailable: False
      include_global_state: True
      partial: False
      wait_for_completion: True
      skip_repo_fs_check: False
      timeout_override:
      continue_if_exception: False
      disable_action: False
    filters:
      - filtertype: none

```

Sample restore action file

Click to expand a sample `action_file.yml` file that runs a restore action.

```

# Remember, leave a key empty if there is no value. None will be a string,
# not a Python "NoneType"
#
# Also remember that all examples have 'disable_action' set to True. If you
# want to use this action as a template, be sure to set this to False after
# copying it.
actions:
  1:
    action: close
    description: "Close selected indices"
    options:
      delete_aliases: False
      timeout_override:
      continue_if_exception: False
      disable_action: False
    filters:
      - filtertype: none
  2:
    action: restore
    description: "restore"
    options:
      repository: datagridbackup
      # Leaving name blank will result in restoring the most recent snapshot by age
      name:
      # Leaving indices blank will result in restoring all indices in the snapshot
      indices:
      include_aliases: False
      ignore_unavailable: False
      include_global_state: True
      partial: False
      rename_pattern:
      rename_replacement:
      extra_settings:
      wait_for_completion: True
      skip_repo_fs_check: False

```

```

timeout_override:
continue_if_exception: False
disable_action: False
filters:
- filtertype: state
state: SUCCESS
exclude: False

```

3.3.2.3 Backup script with email service

You can also use Curator to send an email if an action failed or succeeded.

The script in the example uses a global variable to run and will not work if credentials are needed and it is run outside of the ISE, which contains that global variable. To create a global variable for your ISE session, run the following:

```
$global:cred = get-credential (Run this to create a global variable for your ISE session)
```

Click to expand a sample backup script with email service

Note: Ensure you use the format email@domain.com or the email server will not recognize you.

```

cd "C:\Program Files\elasticsearch-curator"

.\curator.exe --config "C:\Users\csadmin\Desktop\Curator\Configuration.yml" "C:\Users\csadmin\Desktop\Curator\Backup.yml" | Out-File C:\Logs.txt

$Job = (Get-Content C:\Logs.txt | Select-String "successfully completed")
$complete = $Job.Contains("Job completed.")

$Job = (Get-Content C:\Logs.txt | Select-String "Snapshot FAILED")
$failed = $Job.Contains("FAILED")

if($failed) {
    $body = "A Data Grid Backup Job has failed check the backup job or email support@relativity.com to create a ticket"
    Send-MailMessage -From "Jane Smith <jsmith@example.com>"`
        -To "Jane Smith <jsmith@example.com>"`
        -Subject "A Data Grid backup job has failed."`
        -Body $body -BodyAsHTML -SmtpServer smtp.office365.com -credential $cred -UseSsl -
Port 587
}

else {
    $body = "The Data Grid backup job has completed."
    Send-MailMessage -From "Jane Smith <jsmith@example.com>"`
        -To "Jane Smith <jsmith@example.com>"`
        -Subject "A Data Grid backup job has is successful."`
        -Body $body -BodyAsHTML -SmtpServer smtp.office365.com -credential $cred -UseSsl -
Port 587
}

Write-Output("Done")

```

3.3.2.4 Setting the script as a scheduled task

Use the following steps to set the script as a scheduled task:

1. Click **Start > Administrative Tools > Windows Task Scheduler** on the system that runs scheduled tasks.
2. In the **Task Scheduler**, click **Create Task** under **Actions** on the right.
3. Enter a name and description for the task. (Entering a description is optional.)
4. Navigate to the **General** tab, and then select **Security Options**.
5. Specify the user account that runs scheduled tasks. The account can be the same one that runs the Elasticsearch Windows service.
6. Edit the settings to run tasks regardless of whether or not the user is logged in.

The screenshot shows the 'Create New Task' dialog box in Windows Task Scheduler, with the 'General' tab selected. The task name is 'ElasticSearch_Snapshots_Hourly', the location is '\', and the author is 'KIE\''. The description is 'Uses Curator to snapshot all ES indices to the designated backup repository.' Under the 'Security options' section, the user account 'KIE\relservprod' is selected. The 'Run whether user is logged on or not' radio button is selected. Other options include 'Do not store password' (unchecked), 'Run with highest privileges' (checked), and 'Hidden' (unchecked). The 'Configure for' dropdown is set to 'Windows Server 2012 R2'.

7. Navigate to the **Triggers** tab, and then click **New** to add a new trigger for the scheduled task.
8. Verify that the **Begin the Task** field is set to **On a schedule**, and then set the start date to your preferred time.
9. Set the frequency to be every one hour if you're unsure what your recovery point objective goals are.

Note: Relativity stores the last 90 days of audits for each workspace in SQL Server. Long text fields, like extracted text, are usually never edited post import.

10. Set the duration of the task to run indefinitely.
11. Click **OK**.

12. The following example has the task running every hour indefinitely:

The screenshot shows the 'Edit Trigger' dialog box with the following configuration:

- Begin the task:** On a schedule
- Settings:**
 - One time
 - Daily
 - Weekly
 - Monthly
- Start:** 1/ 7/2015 12:00:00 PM
- Synchronize across time zones
- Advanced settings:**
 - Delay task for up to (random delay): 1 hour
 - Repeat task every: 1 hour for a duration of: Indefinitely
 - Stop all running tasks at end of repetition duration
 - Stop task if it runs longer than: 3 days
 - Expire: 1/15/2016 2:19:55 PM
 - Synchronize across time zones
 - Enabled

13. Navigate to the **Actions** tab, and then click **New**.
14. Set the **Action** to **Start a program**.
15. In the **Program/script** field, enter "Powershell."
16. In the **Add arguments (optional)** field, enter the following value:

```
.\[Your PowerShell Script Name]
```

For example, if your PowerShell script is named "Migration1.ps1" then you would enter ".\Migration1.ps1" as the value.

17. In the **Start in (optional)** field, add the location of the folder that contains your PowerShell script. In this example, the script directory is C:\Script. The location entered in the **Start in** box also stores the scheduled task run times, the job history for the copies, and any additional logging that may occur.
18. Click **OK** after configuring your preferred settings.

19. Set any other preferred settings in the **Conditions and Settings** tabs. You can also set up an additional action to email an system admin each time the script runs.
20. Click **OK**.

When you complete these steps, the task runs according to your settings.

3.4 Restoring a snapshot

There are multiple methods for restoring snapshots. Restoring snapshots from the Elasticsearch head console is the recommended procedure, but you can also use cURL to restore snapshots. The Elasticsearch website's documentation relies heavily on cURL commands for snapshot restoration. Brief descriptions of both methods are provided here.

3.4.1 Restoring snapshots from the Elasticsearch head console

You can restore a snapshot from the Elasticsearch head console. Use the following steps to restore a snapshot with this method:

1. Navigate to the Elasticsearch head console URL (http://localhost:9200/_plugin/head).
2. Expand the **Query** tab.
3. Enter the following URL in the first field: **http://localhost:9200/**
4. Enter **_search** in the second field, and use the drop-down menu to select **GET**.
5. Enter the following code to retrieve your snapshot:

```
{
  "type": "fs",
  "settings": {
    "location": "/mount/backups/my_backup",
    "compress": true
  }
}
```

Note: You can restore a snapshot on a functioning cluster, but all indexes residing on the cluster must be closed. The restore only updates closed indexes and creates a new index for any index that doesn't already exist on the cluster.

3.4.2 Restoring snapshots with cURL

Most of the documentation on the Elasticsearch website relies on cURL commands to restore snapshots. You must install cURL for Windows in order to have access to cURL commands in a Windows environment. You can download cURL for Windows .the following site: <http://www.confusedbycode.com/curl/>.

Note: You can paste cURL commands into Marvel Sense (excluding the \$ character), and Marvel automatically converts the cURL command into JSON. The cURL command doesn't convert if typed in manually, you must paste it from your clipboard.

Once you install cURL for Windows, you can restore the cluster state and all indexes in a snapshot with the following cURL command:

```
$ curl -XPOST "localhost:9200/_snapshot/my_backup/snapshot_1/_restore"
```

Note: You can restore a snapshot on a functioning cluster, but all indexes residing on the cluster must be closed. The restore only updates closed indexes and creates a new index for any index that doesn't already exist on the cluster.

For more information on creating or restoring snapshots, see [Snapshot modules](#) on the Elasticsearch website.

4 Data Grid Text Migration application

The Data Grid Text Migration application is an application you can use to migrate your long text fields from SQL to Data Grid.

You can use the Data Grid Text Migration application to migrate your long text fields from SQL to Data Grid. For optimal workspace performance when using SQL features, **we strongly recommend using fixed-length text fields**. Enabling Data Grid allows you to store long text document fields, such as extracted text, outside of the SQL document table. Data Grid utilizes the Relativity file share to store long text document fields, which is the same location where native and image file data is stored.

Note: Once you start a migration job, the results are permanent. You can't delete a migration job once it has been started.

4.1 Special considerations

There are several considerations you should be aware of before and during text migration.

- Minimizing the impact on your SQL document tables, particularly in larger cases with extensive long text document field data.
- Decreasing your SQL footprint or reallocating resources to other workflows in your Relativity instance.
- Storing long text document field data in a separate file share location from your natives and image file data.
- Simple setup and inclusion in your workspace templates before loading data into the workspace.
- Access to the Data Grid Text Migration application, a front-end tool for moving long text document fields in existing workspaces without downtime.

4.1.1 Before running text migration

Before running a text migration job, review the following considerations:

- If your workspace has any custom solutions or third-party applications built on the long text field you're migrating, update them to use the Export API to read long text and metadata fields.
- If your environment is configured for FIPS, you cannot use the Text Migration application.
- If you have any custom data which relies on the long text field you're migrating being stored in the SQL Document table, contact [Relativity Support](#) to discuss alternative solutions before migration.
- Data Grid only supports the **IS SET** and **IS NOT SET** operators. If your workspace uses other operators to query on long text fields, you will need to perform a similar query using dtSearch.
- Mass operations are not supported with Data Grid. If you need to populate a long text field stored in Data Grid, perform an OCR and select the Data Grid field as the destination field.
- Once the text migration job completes, the only index search available for the migrated field is dtSearch. Keyword search is not supported. Data Grid supports all features of dtSearch. If you do not have an active dtSearch index in your workspace, you will need to build one. You can include a com-

bination of Data Grid and SQL fields in your saved search. For more information, see [Running the Breakage Report on page 35](#).

For a list of supported and unsupported Data Grid text functionality, see [Supported and unsupported functionality below](#).

4.1.2 Running text migration

Review the following considerations for running a text migration job:

- The Data Grid Text Migration application runs at the instance level. However, you must install the application to at least one workspace.
- If you have index build in progress (like dtSearch or Analytics), let the index build finish before starting a migration job.
- If you have a mass replace in progress, let the operation finish before starting a migration job.
- We recommend running a migration job in off hours. However, it is not required.
- You can only have one job in progress at a time. If you start a new text migration job while another is in progress, the new job is added to the queue.
- Although the application automatically deletes the column in SQL, you are still required to reclaim the space that was taken up by your long text document field in SQL.

4.1.3 Supported and unsupported functionality


Once you enable a long text field's access to Data Grid, you can't disable it, so it's important to understand the benefits and limitations of storing text in Data Grid.

Supported extracted text functionality	Currently unsupported functionality
<ul style="list-style-type: none">■ Import/export through the Relativity Desktop Client■ Viewer■ Preview■ OCR■ dtSearch indexing and searching■ Persistent highlight sets■ Processing■ Integration Points■ Analytics■ ARM	<ul style="list-style-type: none">■ Keyword search■ SQL queries to long text fields stored in Data Grid■ Adding extracted long fields stored in Data Grid to layouts (including the Document panel)■ RSAPI query■ Pivot and Sort in the UI■ Filtering in the Document list on extracted text■ Mass operations:<ul style="list-style-type: none">○ Edit○ Replace○ Tally/Sum/Average○ Export to File

4.2 Installing the Data Grid Text Migration application

The Data Grid Text Migration application runs at the instance level. However, you must install the application to at least one workspace.

To install the Data Grid Text Migration application:

1. Navigate to the **Application Library** tab.
2. Select **Data Grid Text Migration** from the list of Relativity applications.
3. Next to Workspaces Installed, click **Install**.
4. Next to Workspaces, click , and then select a workspace.
5. Click **Ok**.
6. Click **Save**.

Once you install the Data Grid Text Migration application, the Text Migration Jobs tab appears under the Data Grid tab.

4.2.1 Installing the text migration agents

After you install the Text Migration application, you must install one Data Grid Migration Manager agent and at least one Data Grid Migration Worker agent to your environment. We recommend installing two Data Grid Migration Worker agents.

Note: You can run the Breakage Report without installing the text migration agents. We recommend running the Breakage Report and resolving any issues before running a migration job.

For more information on installing agents, see the Agents guide.

4.3 Running text migration

Once you install the Data Grid Text Migration application and configure your migration agents, you can run text migration. See [Running text migration below](#).

4.4 Running text migration

To run a text migration job, complete the following steps:

1. [Identify workspaces to migrate](#)
2. [Identify long text fields to migrate](#)
3. [Create a text migration job](#)
4. [Run the breakage report](#)
5. [Create and/or activate a dtSearch index](#)

6. [Resolve saved searches and views](#)
7. [Run text migration](#)
8. [Review and resolve migration errors](#)

4.4.1 Identifying workspaces to migrate

Identify the workspace(s) you want to migrate. Be sure to review the special considerations for text migration as well as the list of supported and unsupported functionality for Data Grid. For more information, see [Special considerations on page 31](#).


4.4.2 Identifying long text fields to migrate


Identify which long text field(s) you want to migrate. You can migrate up to three fields per migration job.

4.4.3 Creating a text migration job

To create a text migration job, complete the following:

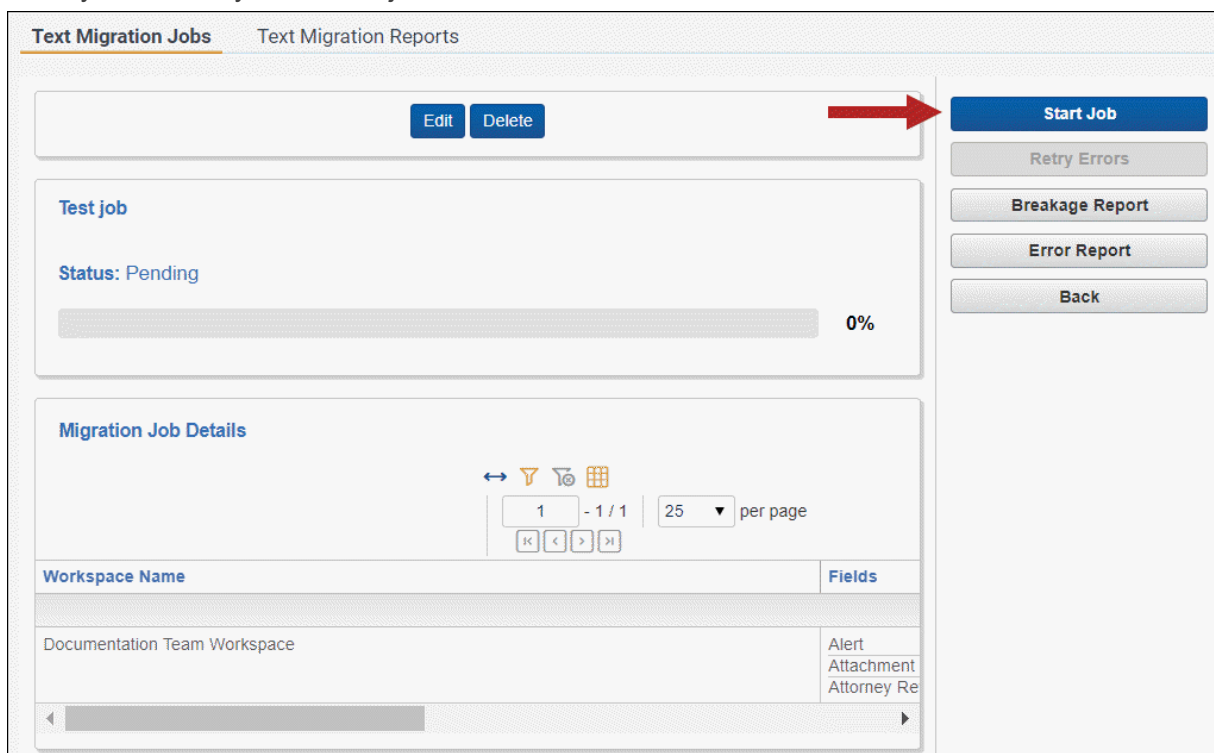
1. Navigate to the **Text Migration Jobs** tab underneath the Data Grid tab.
2. Click **New Text Migration Job**.
3. Complete the following fields:

- **Name** - enter a name for the migration job.
- **Workspaces** - click , and then select the from the list on the left the workspaces you want to migrate. Use the arrows to move your selection(s) to the list on the right. Once you are finished, click **Save**.

- **Fields** - click , and then select the from the list on the left the fields you want to migrate. This list contains all long text fields that aren't Data Grid enabled. Use the arrows to move your selection(s) to the list on the right. Once you are finished, click **Save**.

Note: You can migrate up to three fields per job.

- **Email notification Recipients** - enter the email address(es) of the recipient(s) you want to send a notification to when your migration job finishes running. Separate entries with a semi-colon.
4. Click **Save**. When you first save the job, it has a default status of Pending.
 5. Once you are ready to start the job, click **Start Job** on the console.



4.4.4 Running the Breakage Report

To run the Breakage Report, click **Breakage Report** from the text migration console. The Breakage Report provides a list of all views, saved searches, indexes, and custom objects that will no longer function once you migrate SQL text to Data Grid. We recommend resolving these issues before running a migration job.

One of the primary functions of the Breakage Report is to identify areas where any of the objects above are tied to a keyword search or using keyword search specific functions like Contains or Does Not Contain. Most of the issues identified can be resolved by modifying the object to use an active dtSearch index.

The Breakage Report contains the following columns:

[Back](#)

Breakage Report - Chris's Test Run

The Breakage Report displays Views, Saved Searches, and Custom Objects that will no longer function in the event that you run Chris's Test Run

1 - 3 / 3 10 per page

Workspace Id	Workspace Name	Name	Object Type	Owner	Field
1281017	SQL Workspace - EDRM 500k	Broken 4	Search	Public	Alert
1280873	SQL Workspace 15k - 1	Broken Search 1.3	Search	Martin, Alex	Alert
1280876	SQL Workspace 15k - 4	Broken 4	Search	Public	Alert

- **Workspace ID** - the Artifact ID of the workspace.
- **Workspace Name** - the name of the workspace.
- **Name** - the name of the view, saved search, index, or custom object that will break upon migration.
- **Object Type** - search, view, index, or custom object.
- **Owner** - the owner of the view, saved search, index, or custom object that will break upon migration.
- **Field** - the name of the field causing the object to break.
- **Operator** – the search operator (is, is not, etc.) incompatible with Data Grid.
- **Term** – the text string used in the broken search.

4.4.5 Creating and/or activating a dtSearch index

Most of the issues identified by the breakage report can be resolved by modifying the object to use an active dtSearch index. To create or activate a dtSearch index, see the Searching Guide.

Note: Ensure all fields you want to search on are included in the searchable set for the dtSearch index. You can include a combination of Data Grid and SQL fields in your saved search.

The Data Grid Text Migration application includes logic to automatically fix active dtSearch indexes using the **<all documents in workspace>** searchable set. This logic only applies if the fields being migrated have the Include in Text Index field set to Yes.

The Data Grid Text Migration application creates a saved search called **SS_TextMigration_[JobName]** which contains all fields where Include in Text Index is set to Yes. It also creates a new dtSearch index called **Migration_[existing dtSearch index name]** which uses this new saved search. Once text migration completes, navigate to the old dtSearch index and click **Swap Index**. Select **Migration_[existing dtSearch index name]** as the replacement index.

4.4.6 Resolving saved searches and views

Use the results of the breakage report to help you resolve index or field-level search issues that affect saved searches and views

Note: We recommend copying saved searches prior to making edits.

4.4.6.1 Resolving index searches

1. Navigate to the saved search.
2. Right-click on the name, and then click **Edit**.
3. Click the (Index Search) condition.
4. Copy the terms in the **Search Terms** field.
5. Change the **Index** field to an active dtSearch index.
6. Paste the search terms if needed.
7. Click **Apply**.
8. Click **Save & Search**.

4.4.6.2 Resolving field-level searches

1. Navigate to the saved search.
2. Right-click on the name, and then click **Edit**.
3. Click the condition using the field being migrated (for example, extracted text).
4. Copy the text query, and note the operator (is, is like, contains).
5. Click **Add Condition**, and then select (Index Search).
6. Next to **Index**, select an active dtSearch index.
7. Paste the query text you copied in step 4. Depending on the operator that was used, you may need to convert your search. For more information, see [Search operator conversion below](#).
8. Click **Apply**.
9. Remove the field condition.
10. Click **Save & Search**.

Search operator conversion

Data Grid only supports the **IS SET** and **IS NOT SET** operators. If your field-level search uses another operator, you need to convert the search as follows:

Operator	Example search	Conversion
is	Jane has a broken search	"Jane has a broken search"
is not	Jane has a broken search	NOT "Jane has a broken search"
is set	Supported operator	No modification required

Operator	Example search	Conversion
is not set	Supported operator	No modification required
is less than	N/A	N/A
is greater than	N/A	N/A
is less than or equal to	N/A	N/A
is greater than or equal to	N/A	N/A
is like	Jane has a broken search	*Jane has a broken search*
is not like	Jane has a broken search NOT	*Jane has a broken search*
begins with	Jane has a broken search Paul	Paul*
does not begin with	Jane has a broken search NOT Paul	Paul*
ends with	Jane has a broken search	*broken search
does not end with	Jane has a broken search NOT	*broken search
contains	Jane has a broken search	*Jane has a broken search*
does not contain	Jane has a broken search NOT	*Jane has a broken search*

4.4.7 Running a text migration job

Once you've resolved or taken note of any items in the breakage report, click **Start Job** to start the text migration job. If another text migration job is already in progress, the new job is added to the queue and begins as soon as any running or pending jobs complete.

The job displays one of the following statuses which you can use to monitor the state of your migration:

- Pending
- In Progress
- Complete
- Completed with Errors

The **Migration Job Details** table shows the progress of each workspace in the migration job. This table refreshes every two seconds. Once the job completes or completes with errors, you can filter the table.

The table contains the following columns:

- **Workspace Name** - the name of the workspace being migrated.
- **Fields** - the name of the field being migrated
- **Migration Status** - the status of the field being migrated. The following lists the field migration statuses:
 - Pending
 - In Progress
 - Not In Workspace
 - Already Migrated
 - Completed
 - Completed with Errors

- **Documents Migrated** - the count of documents migrated
- **Total Documents** - the total number of documents migrated.
- **Errors** - the count of documents with errors.

After a successful migration, the application runs a verification step to ensure all documents migrated from SQL to Data Grid. Then, the application drops the SQL column, permanently deleting the migrated text from SQL.

Notes:

- If all documents are migrated, but a workspace and field are still marked as In Progress, this mean they are currently being verified.
- Although the application automatically deletes the column in SQL, you are still required to reclaim the space that was taken up by your long text document field in SQL.

4.4.8 Viewing migration errors

Run the **Data Grid Migration Error Report** from the **Text Migration Reports** tab to view migration errors. To run this report, click **Run**. Click the drop-down to toggle between Field Level Errors and Document Level Errors. Once you've resolved any errors, return to the migration job and click **Retry**. The job status returns to In Progress and any errors on the fields marked with errors are reset.

Data Grid Migration Error Report
 This Relativity script reports on all errors that have occurred while migrating SQL Long Text data into the Data Grid repository.

Buttons: **Preview** **Run**

Document Level Errors (selected) | Field Level Errors | Items: 1 - 13 (of 13)

#	Field Level Errors	Workspace Id	Workspace Name	Field Name	Document ID	Error Message
1	Text Load test - Text Migration	1298214	EDRM SQL Ex Text - Text Migration Load Testing	Extracted Text	3319736	- DataGrid.Exceptions.DataGr Error writing documents to Data Grid file system Response Result: ";3c85b069-e90f-4f1f- 9ee3- 7d312d2be388','Error'," 'Fields.ExtractedText','Error', network path was not found.' at DataGrid.Implementations.D <WriteRecords>d__6.MoveN --- End of stack trace from previous location where exception was thrown --- at System.Runtime.ExceptionS at System.Runtime.CompilerSe task) at DataGrid.Implementations.Fi

Footer: All 13 | Export to File | Go | Viewing all items in sets of 25 per page

4.4.8.1 Field Level Errors

The following results populate the bottom of the window:

- **Job Name** - the name of the migration job where the errors occurred.
- **Workspace Id** - the artifact ID of the workspace.
- **Workspace Name** - the name of the workspace.
- **Field Id** - the Artifact ID of the field with errors.
- **Field Name** - the name of the field with errors.
- **Error Message** - the text of the error.

4.4.8.2 Document Level Errors

The following results populate the bottom of the window:

- **Job Name** - the name of the migration job where the errors occurred.
- **Workspace Id** - the artifact ID of the workspace.
- **Workspace Name** - the name of the workspace.
- **Document ID** - the artifact ID of the document.
- **Field Name** - the name of the field with errors.
- **Error Message** - the text of the error.

Proprietary Rights

This documentation (“**Documentation**”) and the software to which it relates (“**Software**”) belongs to Relativity ODA LLC and/or Relativity’s third party software vendors. Relativity grants written license agreements which contain restrictions. All parties accessing the Documentation or Software must: respect proprietary rights of Relativity and third parties; comply with your organization’s license agreement, including but not limited to license restrictions on use, copying, modifications, reverse engineering, and derivative products; and refrain from any misuse or misappropriation of this Documentation or Software in whole or in part. The Software and Documentation is protected by the **Copyright Act of 1976**, as amended, and the Software code is protected by the **Illinois Trade Secrets Act**. Violations can involve substantial civil liabilities, exemplary damages, and criminal penalties, including fines and possible imprisonment.

©2024. Relativity ODA LLC. All rights reserved. Relativity® is a registered trademark of Relativity ODA LLC.