Known issues

This article summarizes known issues and workarounds for reported issues.

Some tasks are incompatible with MariaDB

Applies to:

Raynet One Data Hub 14.1 with MariaDB backend

Description:

The following tasks are available for Microsoft SQL Server only:

- 1. Data Collection → Intune → (2 SQL Tasks)
- 3. Data Analysis → Software Asset Management → Database → Oracle
- 1. Data Collection → ITSM → Import → RayVentory SNMP Devices
- 1. Data Collection → Inventory → ECM WMI → ECM WMI

Workaround:

Currently none, the SQL query can be manually converted to be MariaDB compatible

---

Data Collection → Inventory → RayVentory dashboard have wrong sorting

Applies to:

Raynet One Data Hub 14.1 with MariaDB backend

Description:

- 1. Data Collection → Inventory → RayVentory - The Inventory Dates chart in the bottom right have incorrect sorting.

Workaround:

Currently none.

Data Analysis → IT Visibility tasks are not performing properly

Applies to:

Raynet One Data Hub 14.1 with MariaDB backend

Description:

Tasks in the Data Analysis → IT Visibility folder may be slower than their SQL Server counterparts due to a less powerful query analyzer.

Workaround:

If IT Visibility task performance is low, the following code, added to the beginning of IT Visibility tasks, can improve their performance:

```
SELECT COUNT(*) INTO @index_exists
FROM information_schema.statistics
WHERE table_schema = DATABASE()
AND table_name = 'DataTransformation-result_devices'
AND index_name = 'device_import_id_idx';

-- Create index only if it does not exist
IF @index_exists = 0 THEN
    CREATE INDEX device_import_id_idx ON `DataTransformation-result_devices` (import_id);
END IF;

SELECT COUNT(*) INTO @index_exists
FROM information_schema.statistics
WHERE table_schema = DATABASE()
AND table_name = 'DataTransformation-result_software_summary'
AND index_name = 'software_id_idx';

-- Create index only if it does not exist
IF @index_exists = 0 THEN
    CREATE INDEX software_id_idx ON `DataTransformation-result_software_summary` (Id);
END IF;
```

Missing table error for EOL/EOS tasks

Applies to:

Raynet One Data Hub 14.1 with MariaDB backend

Description:

Some users have reported that their EOL/EOS tasks are not working properly - the execution terminates with a missing table error message.

Workaround:

Replace the contents of the SQL code with the following:

```
SELECT
 COUNT(DISTINCT `dtrd`.`device_name`) AS `device_name_Count`,
 `cs`.`vendor` as Vendor,
 `cs`.`name` as Name,
 `cs`.`version` as Version,
 `csf`.`title` AS `Functionality`,
 `cs`.`EndOfLifeDate`,
 `cs`.`Support`,
 `cs`.`ExtendedSupportDate`
FROM `Catalog_v2_SoftwareOnDeviceMatching` `csm`
INNER JOIN `DataTransformation-result_devices` `dtrd` ON `dtrd`.`import_id` = `csm`.`computerid`
INNER JOIN `Catalog_v2-Software` `cs` ON `cs`.`softwareid` = `csm`.`softwareid`
LEFT JOIN `Catalog_v2-SoftwareFunctionMatching` `csfm` ON `csfm`.`softwareid` = `cs`.`softwareid`
```

```
LEFT JOIN `Catalog_v2-SoftwareFunction` `csf` ON `csf`.`functionid` = `csfm`.`functionid`
WHERE `name` IS NOT NULL
GROUP BY
 `cs`.`vendor`,
 `cs`.`name`,
 `cs`.`version`,
 `csf`.`title`,
 `cs`.`endoflifedate`,
 `cs`.`support`,
 `cs`.`extendedsupportdate`;
```

“GuestOS column does not exist" error in Hypervisor Dashboard Overview

Applies to:
Raynet One Data Hub 14.1

Description:
The dashboard Hypervisor Dashboard Overview throws an error about a missing column. The name of the missing column is `GuestOS`.

Workaround:
Replace the following part in the SQL code belonging to the task:

```
`hr`.`GuestOS` AS `GuestOS`,
```

with this:

```
`hr`.`GuestOS` AS `OS`,
```

Raynet One MSSQL / MariaDB direct tasks throw "Sequence contains no elements" error

Applies to:
Raynet One Data Hub 14.1 and direct connection to the Raynet One database.

Description:
Running MSSQL & MariaDB Raynet One tasks causes transformation to throw "Sequence contains no elements" error.

Workaround:
Raynet One data should only be queried via the dedicated API. Remove the direct query source from the transformation step and migrate to the API usage.

Error “SQLite Error 1” when executing the transformation task

Applies to:
Raynet One Data Hub 14.1, default data transformation process with incomplete variable set

Description:
When executing the transformation the following error may be returned:

```
Transform active directory device data' (Map#101): SQLite Error 1: 'near ")": syntax error'.
```

This problem only occurs on systems where the value of the `LAST_SEEN_DAYS` property is missing.

Workaround:
Set the value of the `LAST_SEEN_DAYS` property to a non-empty text (number).

Default database is not created (Docker + MSSQL)

Applies to:

Raynet One Data Hub 14.1 in Docker with MSSQL backend

Description:

Some users reported that the default database, `DataHub_Default`, is not properly initialized when the Docker container set is started for the first time.

Workaround:

Manually create the `DataHub_Default` database before or after starting the instance.

Shared links not working after migration

Applies to:

Raynet One Data Hub 14.1

Description:

After migrating from version 14.0, previously shared dashboards and reports do not work.

Workaround:

Re-create each shared link for the dashboards and reports to be shared.

Searching for tasks only applies to the current folder

Applies to:

Raynet One Data Hub 14.1

Description:

When searching for a task using the top right search box, only the results from the currently viewed folder are returned. In the previous version, the search was global.

Workaround:

Currently none, this will be fixed in the next minor update.

Slow performance of AI-assistant

Applies to:

Raynet One Data Hub 14.1 with AI features enabled and MariaDB backend

Description:

Some queries and prompts may take a long time to complete. Since the AI assistant works on the available data, it shares the same performance bottleneck as the underlying dataset. This can be observed when working with large data sets and when prompting for an answer that performs heavy grouping, merging, etc., especially on unindexed columns. This issue applies to MariaDB backend. Microsoft SQL Server seem to perform well in most demanding scenarios.

Workaround:

Use simpler prompts to avoid excessive data operations. Create indexes in strategic places so that data access is fast. For complex data sets, consider switching to MS SQL Server.

Citrix Director Devices on MariaDB

Applies to:

Raynet One Data Hub 14.1 with MariaDB backend

Description:

When saving received data in the MariaDB database, the following error occurs:

"Row size too large. The maximum row size for the used table type, excluding BLOBs, is 65,535 bytes. This includes storage overhead. Refer to the MariaDB manual for details. To resolve this, some columns need to be changed to TEXT or BLOB."

Workaround:

Replace the query for the task **"Citrix Director Devices"** with the following:

```
SELECT DISTINCT
  LEFT(REPLACE([Name],' ','_'), 255) AS import_id,
  'RayVentory' AS import_data_source_id,
  LEFT(REPLACE([Name],' ','_'), 255) AS device_key,
  'unknown_costcenter' AS import_org_level_2_id,
  LEFT(REPLACE([Name],' ','_'), 255) AS device_name,
  NULL AS device_manufacturer,
  NULL AS device_model,
  NULL AS serial_number,
  NULL AS import_operating_system_id,
  'xen_cluster' AS import_device_type_id,
  'Active' AS import_device_status_id,
  NULL AS import_domain_id,
  NULL AS import_user_id,
  NULL AS ip_address,
  NULL AS mac_address,
  LEFT(REPLACE([Name],' ','_'), 255) AS fqdn,
  NULL AS inventory_date,
  NULL AS installation_date,
  NULL AS cpu_chip_count,
  NULL AS cpu_core_count,
  NULL AS cpu_speed,
  NULL AS import_cpu_type_id,
  NULL AS ram,
  NULL AS storage,
  NULL AS graphics,
  NULL AS network,
  NULL AS bios,
  NULL AS cpu_core_count_limited
FROM MonitorData.Hypervisor

UNION ALL

SELECT DISTINCT
  LEFT(DnsName, 255) AS import_id,
  'RayVentory' AS import_data_source_id,
  LEFT(DnsName, 255) AS device_key,
  'unknown_costcenter' AS import_org_level_2_id,
  LEFT(HostedMachineName, 255) AS device_name,
  'Citrix Systems, Inc.' AS device_manufacturer,
  'Citrix XenDesktop' AS device_model,
  [Hash] AS serial_number,
  OSType AS import_operating_system_id,
  'virtual_server' AS import_device_type_id,
  'Active' AS import_device_status_id,
```

```
  LEFT(SUBSTRING(DnsName, CHARINDEX('.',DnsName,0)+1, LEN(DnsName)-CHARINDEX('.',DnsName,0)), 255) AS import_domain_id,
  NULL AS import_user_id,
  NULL AS ip_address,
  NULL AS mac_address,
  LEFT(DnsName, 255) AS fqdn,
  NULL AS inventory_date,
  NULL AS installation_date,
  NULL AS cpu_chip_count,
  NULL AS cpu_core_count,
  NULL AS cpu_speed,
  NULL AS import_cpu_type_id,
  NULL AS ram,
  NULL AS storage,
  NULL AS graphics,
  NULL AS network,
  NULL AS bios,
  NULL AS cpu_core_count_limited
FROM MonitorData.Machine
WHERE ISNULL(DnsName ,'') <> ''

UNION ALL

SELECT DISTINCT
  LEFT(HostingServerName, 255) AS import_id,
  'RayVentory' AS import_data_source_id,
  LEFT(HostingServerName, 255) AS device_key,
  'unknown_costcenter' AS import_org_level_2_id,
  LEFT(HostingServerName, 255) AS device_name,
  NULL AS device_manufacturer,
  NULL AS device_model,
  NULL AS serial_number,
  NULL AS import_operating_system_id,
  'xen_host' AS import_device_type_id,
  'Active' AS import_device_status_id,
  LEFT(SUBSTRING(HostingServerName, CHARINDEX('.',HostingServerName,0)+1, LEN(HostingServerName)-CHARINDEX('.',HostingServerName,0)), 255) AS import_domain_id,
  NULL AS import_user_id,
  NULL AS ip_address,
  NULL AS mac_address,
  LEFT(HostingServerName, 255) AS fqdn,
  NULL AS inventory_date,
  NULL AS installation_date,
  NULL AS cpu_chip_count,
  NULL AS cpu_core_count,
  NULL AS cpu_speed,
  NULL AS import_cpu_type_id,
  NULL AS ram,
  NULL AS storage,
  NULL AS graphics,
  NULL AS network,
  NULL AS bios,
  NULL AS cpu_core_count_limited
FROM MonitorData.Machine
WHERE ISNULL(HostingServerName,'') <> ''
```

Upgrading from 14.0 Update 1 to Data Hub 14.1 could fail under certain conditions and need manual execution of SQL script

Applies to:

Raynet One Data Hub 14.1 upgraded from 14.0 [Update 1] with MSSQL backend.

Description:

The migration using MSSQL in Docker or via MSI with Data Hub version 14.0 [Update 1] to Data Hub 14.1 could fail in certain conditions and will need execution of SQL migration script manually.

Workaround:

Execute this migration script manually against your MSSQL database:

```
PRINT N'Add quartz tables';
DROP INDEX IF EXISTS [IDX_QRTZ_T_J] ON [dbo].[QRTZ_TRIGGERS];
DROP INDEX IF EXISTS [IDX_QRTZ_T_JG] ON [dbo].[QRTZ_TRIGGERS];
DROP INDEX IF EXISTS [IDX_QRTZ_T_C] ON [dbo].[QRTZ_TRIGGERS];
DROP INDEX IF EXISTS [IDX_QRTZ_T_G] ON [dbo].[QRTZ_TRIGGERS];
DROP INDEX IF EXISTS [IDX_QRTZ_T_G_J] ON [dbo].[QRTZ_TRIGGERS];
DROP INDEX IF EXISTS [IDX_QRTZ_T_STATE] ON [dbo].[QRTZ_TRIGGERS];
DROP INDEX IF EXISTS [IDX_QRTZ_T_N_STATE] ON [dbo].[QRTZ_TRIGGERS];
DROP INDEX IF EXISTS [IDX_QRTZ_T_N_G_STATE] ON [dbo].[QRTZ_TRIGGERS];
DROP INDEX IF EXISTS [IDX_QRTZ_T_NEXT_FIRE_TIME] ON [dbo].[QRTZ_TRIGGERS];
DROP INDEX IF EXISTS [IDX_QRTZ_T_NFT_ST] ON [dbo].[QRTZ_TRIGGERS];
DROP INDEX IF EXISTS [IDX_QRTZ_T_NFT_MISFIRE] ON [dbo].[QRTZ_TRIGGERS];
DROP INDEX IF EXISTS [IDX_QRTZ_T_NFT_ST_MISFIRE] ON [dbo].[QRTZ_TRIGGERS];
DROP INDEX IF EXISTS [IDX_QRTZ_T_NFT_ST_MISFIRE_GRP] ON [dbo].[QRTZ_TRIGGERS];
DROP INDEX IF EXISTS [IDX_QRTZ_FT_TRIG_INST_NAME] ON [dbo].[QRTZ_FIRED_TRIGGERS];
DROP INDEX IF EXISTS [IDX_QRTZ_FT_INST_JOB_REQ_RCVRY] ON [dbo].[QRTZ_FIRED_TRIGGERS];
DROP INDEX IF EXISTS [IDX_QRTZ_FT_J_G] ON [dbo].[QRTZ_FIRED_TRIGGERS];
DROP INDEX IF EXISTS [IDX_QRTZ_FT_JG] ON [dbo].[QRTZ_FIRED_TRIGGERS];
DROP INDEX IF EXISTS [IDX_QRTZ_FT_T_G] ON [dbo].[QRTZ_FIRED_TRIGGERS];
DROP INDEX IF EXISTS [IDX_QRTZ_FT_TG] ON [dbo].[QRTZ_FIRED_TRIGGERS];
DROP INDEX IF EXISTS [IDX_QRTZ_FT_G_J] ON [dbo].[QRTZ_FIRED_TRIGGERS];
DROP INDEX IF EXISTS [IDX_QRTZ_FT_G_T] ON [dbo].[QRTZ_FIRED_TRIGGERS];

IF OBJECT_ID(N'[dbo].[FK_QRTZ_TRIGGERS_QRTZ_JOB_DETAILS]', N'F') IS NOT NULL
ALTER TABLE [dbo].[QRTZ_TRIGGERS] DROP CONSTRAINT [FK_QRTZ_TRIGGERS_QRTZ_JOB_DETAILS];

IF OBJECT_ID(N'[dbo].[FK_QRTZ_CRON_TRIGGERS_QRTZ_TRIGGERS]', N'F') IS NOT NULL
ALTER TABLE [dbo].[QRTZ_CRON_TRIGGERS] DROP CONSTRAINT [FK_QRTZ_CRON_TRIGGERS_QRTZ_TRIGGERS];

IF OBJECT_ID(N'[dbo].[FK_QRTZ_SIMPLE_TRIGGERS_QRTZ_TRIGGERS]', N'F') IS NOT NULL
ALTER TABLE [dbo].[QRTZ_SIMPLE_TRIGGERS] DROP CONSTRAINT [FK_QRTZ_SIMPLE_TRIGGERS_QRTZ_TRIGGERS];

IF OBJECT_ID(N'[dbo].[FK_QRTZ_SIMPROP_TRIGGERS_QRTZ_TRIGGERS]', N'F') IS NOT NULL
ALTER TABLE [dbo].[QRTZ_SIMPROP_TRIGGERS] DROP CONSTRAINT [FK_QRTZ_SIMPROP_TRIGGERS_QRTZ_TRIGGERS];

IF  EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id = OBJECT_ID(N'[dbo].
[FK_QRTZ_JOB_LISTENERS_QRTZ_JOB_DETAILS]') AND parent_object_id = OBJECT_ID(N'[dbo].[QRTZ_JOB_LISTENERS]'))
ALTER TABLE [dbo].[QRTZ_JOB_LISTENERS] DROP CONSTRAINT [FK_QRTZ_JOB_LISTENERS_QRTZ_JOB_DETAILS];

IF  EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id = OBJECT_ID(N'[dbo].
[FK_QRTZ_TRIGGER_LISTENERS_QRTZ_TRIGGERS]') AND parent_object_id = OBJECT_ID(N'[dbo].
[QRTZ_TRIGGER_LISTENERS]'))
ALTER TABLE [dbo].[QRTZ_TRIGGER_LISTENERS] DROP CONSTRAINT [FK_QRTZ_TRIGGER_LISTENERS_QRTZ_TRIGGERS];

IF OBJECT_ID(N'[dbo].[QRTZ_CALENDARS]', N'U') IS NOT NULL
```

```
DROP TABLE [dbo].[QRTZ_CALENDARS];

IF OBJECT_ID(N'[dbo].[QRTZ_CRON_TRIGGERS]', N'U') IS NOT NULL
DROP TABLE [dbo].[QRTZ_CRON_TRIGGERS];

IF OBJECT_ID(N'[dbo].[QRTZ_BLOB_TRIGGERS]', N'U') IS NOT NULL
DROP TABLE [dbo].[QRTZ_BLOB_TRIGGERS];

IF OBJECT_ID(N'[dbo].[QRTZ_FIRED_TRIGGERS]', N'U') IS NOT NULL
DROP TABLE [dbo].[QRTZ_FIRED_TRIGGERS];

IF OBJECT_ID(N'[dbo].[QRTZ_PAUSED_TRIGGER_GRPS]', N'U') IS NOT NULL
DROP TABLE [dbo].[QRTZ_PAUSED_TRIGGER_GRPS];

IF  OBJECT_ID(N'[dbo].[QRTZ_JOB_LISTENERS]', N'U') IS NOT NULL
DROP TABLE [dbo].[QRTZ_JOB_LISTENERS];

IF OBJECT_ID(N'[dbo].[QRTZ_SCHEDULER_STATE]', N'U') IS NOT NULL
DROP TABLE [dbo].[QRTZ_SCHEDULER_STATE];

IF OBJECT_ID(N'[dbo].[QRTZ_LOCKS]', N'U') IS NOT NULL
DROP TABLE [dbo].[QRTZ_LOCKS];

IF OBJECT_ID(N'[dbo].[QRTZ_TRIGGER_LISTENERS]', N'U') IS NOT NULL
DROP TABLE [dbo].[QRTZ_TRIGGER_LISTENERS];

IF OBJECT_ID(N'[dbo].[QRTZ_JOB_DETAILS]', N'U') IS NOT NULL
DROP TABLE [dbo].[QRTZ_JOB_DETAILS];

IF OBJECT_ID(N'[dbo].[QRTZ_SIMPLE_TRIGGERS]', N'U') IS NOT NULL
DROP TABLE [dbo].[QRTZ_SIMPLE_TRIGGERS];

IF OBJECT_ID(N'[dbo].[QRTZ_SIMPROP_TRIGGERS]', N'U') IS NOT NULL
DROP TABLE [dbo].[QRTZ_SIMPROP_TRIGGERS];

IF OBJECT_ID(N'[dbo].[QRTZ_TRIGGERS]', N'U') IS NOT NULL
DROP TABLE [dbo].[QRTZ_TRIGGERS];

CREATE TABLE [dbo].[QRTZ_CALENDARS] (
  [SCHED_NAME] nvarchar(120) NOT NULL,
  [CALENDAR_NAME] nvarchar(200) NOT NULL,
  [CALENDAR] varbinary(max) NOT NULL
);

CREATE TABLE [dbo].[QRTZ_CRON_TRIGGERS] (
  [SCHED_NAME] nvarchar(120) NOT NULL,
  [TRIGGER_NAME] nvarchar(150) NOT NULL,
  [TRIGGER_GROUP] nvarchar(150) NOT NULL,
  [CRON_EXPRESSION] nvarchar(120) NOT NULL,
  [TIME_ZONE_ID] nvarchar(80)
);

CREATE TABLE [dbo].[QRTZ_FIRED_TRIGGERS] (
  [SCHED_NAME] nvarchar(120) NOT NULL,
  [ENTRY_ID] nvarchar(140) NOT NULL,
  [TRIGGER_NAME] nvarchar(150) NOT NULL,
  [TRIGGER_GROUP] nvarchar(150) NOT NULL,
  [INSTANCE_NAME] nvarchar(200) NOT NULL,
```

```
  [FIRED_TIME] bigint NOT NULL,
  [SCHED_TIME] bigint NOT NULL,
  [PRIORITY] int NOT NULL,
  [STATE] nvarchar(16) NOT NULL,
  [JOB_NAME] nvarchar(150) NULL,
  [JOB_GROUP] nvarchar(150) NULL,
  [IS_NONCONCURRENT] bit NULL,
  [REQUESTS_RECOVERY] bit NULL
);

CREATE TABLE [dbo].[QRTZ_PAUSED_TRIGGER_GRPS] (
  [SCHED_NAME] nvarchar(120) NOT NULL,
  [TRIGGER_GROUP] nvarchar(150) NOT NULL
);

CREATE TABLE [dbo].[QRTZ_SCHEDULER_STATE] (
  [SCHED_NAME] nvarchar(120) NOT NULL,
  [INSTANCE_NAME] nvarchar(200) NOT NULL,
  [LAST_CHECKIN_TIME] bigint NOT NULL,
  [CHECKIN_INTERVAL] bigint NOT NULL
);

CREATE TABLE [dbo].[QRTZ_LOCKS] (
  [SCHED_NAME] nvarchar(120) NOT NULL,
  [LOCK_NAME] nvarchar(40) NOT NULL
);

CREATE TABLE [dbo].[QRTZ_JOB_DETAILS] (
  [SCHED_NAME] nvarchar(120) NOT NULL,
  [JOB_NAME] nvarchar(150) NOT NULL,
  [JOB_GROUP] nvarchar(150) NOT NULL,
  [DESCRIPTION] nvarchar(250) NULL,
  [JOB_CLASS_NAME] nvarchar(250) NOT NULL,
  [IS_DURABLE] bit NOT NULL,
  [IS_NONCONCURRENT] bit NOT NULL,
  [IS_UPDATE_DATA] bit NOT NULL,
  [REQUESTS_RECOVERY] bit NOT NULL,
  [JOB_DATA] varbinary(max) NULL
);

CREATE TABLE [dbo].[QRTZ_SIMPLE_TRIGGERS] (
  [SCHED_NAME] nvarchar(120) NOT NULL,
  [TRIGGER_NAME] nvarchar(150) NOT NULL,
  [TRIGGER_GROUP] nvarchar(150) NOT NULL,
  [REPEAT_COUNT] int NOT NULL,
  [REPEAT_INTERVAL] bigint NOT NULL,
  [TIMES_TRIGGERED] int NOT NULL
);

CREATE TABLE [dbo].[QRTZ_SIMPROP_TRIGGERS] (
  [SCHED_NAME] nvarchar(120) NOT NULL,
  [TRIGGER_NAME] nvarchar(150) NOT NULL,
  [TRIGGER_GROUP] nvarchar(150) NOT NULL,
  [STR_PROP_1] nvarchar(512) NULL,
  [STR_PROP_2] nvarchar(512) NULL,
  [STR_PROP_3] nvarchar(512) NULL,
  [INT_PROP_1] int NULL,
  [INT_PROP_2] int NULL,
```

```
  [LONG_PROP_1] bigint NULL,
  [LONG_PROP_2] bigint NULL,
  [DEC_PROP_1] numeric(13,4) NULL,
  [DEC_PROP_2] numeric(13,4) NULL,
  [BOOL_PROP_1] bit NULL,
  [BOOL_PROP_2] bit NULL,
  [TIME_ZONE_ID] nvarchar(80) NULL
);

CREATE TABLE [dbo].[QRTZ_BLOB_TRIGGERS] (
  [SCHED_NAME] nvarchar(120) NOT NULL,
  [TRIGGER_NAME] nvarchar(150) NOT NULL,
  [TRIGGER_GROUP] nvarchar(150) NOT NULL,
  [BLOB_DATA] varbinary(max) NULL
);

CREATE TABLE [dbo].[QRTZ_TRIGGERS] (
  [SCHED_NAME] nvarchar(120) NOT NULL,
  [TRIGGER_NAME] nvarchar(150) NOT NULL,
  [TRIGGER_GROUP] nvarchar(150) NOT NULL,
  [JOB_NAME] nvarchar(150) NOT NULL,
  [JOB_GROUP] nvarchar(150) NOT NULL,
  [DESCRIPTION] nvarchar(250) NULL,
  [NEXT_FIRE_TIME] bigint NULL,
  [PREV_FIRE_TIME] bigint NULL,
  [PRIORITY] int NULL,
  [TRIGGER_STATE] nvarchar(16) NOT NULL,
  [TRIGGER_TYPE] nvarchar(8) NOT NULL,
  [START_TIME] bigint NOT NULL,
  [END_TIME] bigint NULL,
  [CALENDAR_NAME] nvarchar(200) NULL,
  [MISFIRE_INSTR] int NULL,
  [JOB_DATA] varbinary(max) NULL
);
GO

ALTER TABLE [dbo].[QRTZ_CALENDARS] WITH NOCHECK ADD
  CONSTRAINT [PK_QRTZ_CALENDARS] PRIMARY KEY  CLUSTERED
  (
    [SCHED_NAME],
    [CALENDAR_NAME]
  );
GO

ALTER TABLE [dbo].[QRTZ_CRON_TRIGGERS] WITH NOCHECK ADD
  CONSTRAINT [PK_QRTZ_CRON_TRIGGERS] PRIMARY KEY  CLUSTERED
  (
    [SCHED_NAME],
    [TRIGGER_NAME],
    [TRIGGER_GROUP]
  );
GO

ALTER TABLE [dbo].[QRTZ_FIRED_TRIGGERS] WITH NOCHECK ADD
  CONSTRAINT [PK_QRTZ_FIRED_TRIGGERS] PRIMARY KEY  CLUSTERED
  (
    [SCHED_NAME],
    [ENTRY_ID]
```

```
  );
GO

ALTER TABLE [dbo].[QRTZ_PAUSED_TRIGGER_GRPS] WITH NOCHECK ADD
  CONSTRAINT [PK_QRTZ_PAUSED_TRIGGER_GRPS] PRIMARY KEY  CLUSTERED
  (
    [SCHED_NAME],
    [TRIGGER_GROUP]
  );
GO

ALTER TABLE [dbo].[QRTZ_SCHEDULER_STATE] WITH NOCHECK ADD
  CONSTRAINT [PK_QRTZ_SCHEDULER_STATE] PRIMARY KEY  CLUSTERED
  (
    [SCHED_NAME],
    [INSTANCE_NAME]
  );
GO

ALTER TABLE [dbo].[QRTZ_LOCKS] WITH NOCHECK ADD
  CONSTRAINT [PK_QRTZ_LOCKS] PRIMARY KEY  CLUSTERED
  (
    [SCHED_NAME],
    [LOCK_NAME]
  );
GO

ALTER TABLE [dbo].[QRTZ_JOB_DETAILS] WITH NOCHECK ADD
  CONSTRAINT [PK_QRTZ_JOB_DETAILS] PRIMARY KEY  CLUSTERED
  (
    [SCHED_NAME],
    [JOB_NAME],
    [JOB_GROUP]
  );
GO

ALTER TABLE [dbo].[QRTZ_SIMPLE_TRIGGERS] WITH NOCHECK ADD
  CONSTRAINT [PK_QRTZ_SIMPLE_TRIGGERS] PRIMARY KEY  CLUSTERED
  (
    [SCHED_NAME],
    [TRIGGER_NAME],
    [TRIGGER_GROUP]
  );
GO

ALTER TABLE [dbo].[QRTZ_SIMPROP_TRIGGERS] WITH NOCHECK ADD
  CONSTRAINT [PK_QRTZ_SIMPROP_TRIGGERS] PRIMARY KEY  CLUSTERED
  (
    [SCHED_NAME],
    [TRIGGER_NAME],
    [TRIGGER_GROUP]
  );
GO

ALTER TABLE [dbo].[QRTZ_TRIGGERS] WITH NOCHECK ADD
  CONSTRAINT [PK_QRTZ_TRIGGERS] PRIMARY KEY  CLUSTERED
  (
    [SCHED_NAME],
```

```
    [TRIGGER_NAME],
    [TRIGGER_GROUP]
  );
GO

ALTER TABLE [dbo].[QRTZ_BLOB_TRIGGERS] WITH NOCHECK ADD
  CONSTRAINT [PK_QRTZ_BLOB_TRIGGERS] PRIMARY KEY  CLUSTERED
  (
    [SCHED_NAME],
    [TRIGGER_NAME],
    [TRIGGER_GROUP]
  );
GO

ALTER TABLE [dbo].[QRTZ_CRON_TRIGGERS] ADD
  CONSTRAINT [FK_QRTZ_CRON_TRIGGERS_QRTZ_TRIGGERS] FOREIGN KEY
  (
    [SCHED_NAME],
    [TRIGGER_NAME],
    [TRIGGER_GROUP]
  ) REFERENCES [dbo].[QRTZ_TRIGGERS] (
    [SCHED_NAME],
    [TRIGGER_NAME],
    [TRIGGER_GROUP]
  ) ON DELETE CASCADE;
GO

ALTER TABLE [dbo].[QRTZ_SIMPLE_TRIGGERS] ADD
  CONSTRAINT [FK_QRTZ_SIMPLE_TRIGGERS_QRTZ_TRIGGERS] FOREIGN KEY
  (
    [SCHED_NAME],
    [TRIGGER_NAME],
    [TRIGGER_GROUP]
  ) REFERENCES [dbo].[QRTZ_TRIGGERS] (
    [SCHED_NAME],
    [TRIGGER_NAME],
    [TRIGGER_GROUP]
  ) ON DELETE CASCADE;
GO

ALTER TABLE [dbo].[QRTZ_SIMPROP_TRIGGERS] ADD
  CONSTRAINT [FK_QRTZ_SIMPROP_TRIGGERS_QRTZ_TRIGGERS] FOREIGN KEY
  (
    [SCHED_NAME],
    [TRIGGER_NAME],
    [TRIGGER_GROUP]
  ) REFERENCES [dbo].[QRTZ_TRIGGERS] (
    [SCHED_NAME],
    [TRIGGER_NAME],
    [TRIGGER_GROUP]
  ) ON DELETE CASCADE;
GO

ALTER TABLE [dbo].[QRTZ_TRIGGERS] ADD
  CONSTRAINT [FK_QRTZ_TRIGGERS_QRTZ_JOB_DETAILS] FOREIGN KEY
  (
    [SCHED_NAME],
    [JOB_NAME],
```

```
    [JOB_GROUP]
  ) REFERENCES [dbo].[QRTZ_JOB_DETAILS] (
    [SCHED_NAME],
    [JOB_NAME],
    [JOB_GROUP]
  );
GO

-- Create indexes
CREATE INDEX [IDX_QRTZ_T_G_J]                ON [dbo].[QRTZ_TRIGGERS](SCHED_NAME, JOB_GROUP, JOB_NAME);
CREATE INDEX [IDX_QRTZ_T_C]                  ON [dbo].[QRTZ_TRIGGERS](SCHED_NAME, CALENDAR_NAME);

CREATE INDEX [IDX_QRTZ_T_N_G_STATE]          ON [dbo].[QRTZ_TRIGGERS](SCHED_NAME, TRIGGER_GROUP,
TRIGGER_STATE);
CREATE INDEX [IDX_QRTZ_T_STATE]              ON [dbo].[QRTZ_TRIGGERS](SCHED_NAME, TRIGGER_STATE);
CREATE INDEX [IDX_QRTZ_T_N_STATE]            ON [dbo].[QRTZ_TRIGGERS](SCHED_NAME, TRIGGER_NAME,
TRIGGER_GROUP, TRIGGER_STATE);
CREATE INDEX [IDX_QRTZ_T_NEXT_FIRE_TIME]     ON [dbo].[QRTZ_TRIGGERS](SCHED_NAME, NEXT_FIRE_TIME);
CREATE INDEX [IDX_QRTZ_T_NFT_ST]             ON [dbo].[QRTZ_TRIGGERS](SCHED_NAME, TRIGGER_STATE,
NEXT_FIRE_TIME);
CREATE INDEX [IDX_QRTZ_T_NFT_ST_MISFIRE]     ON [dbo].[QRTZ_TRIGGERS](SCHED_NAME, MISFIRE_INSTR,
NEXT_FIRE_TIME, TRIGGER_STATE);
CREATE INDEX [IDX_QRTZ_T_NFT_ST_MISFIRE_GRP] ON [dbo].[QRTZ_TRIGGERS](SCHED_NAME, MISFIRE_INSTR,
NEXT_FIRE_TIME, TRIGGER_GROUP, TRIGGER_STATE);

CREATE INDEX [IDX_QRTZ_FT_INST_JOB_REQ_RCVRY] ON [dbo].[QRTZ_FIRED_TRIGGERS](SCHED_NAME, INSTANCE_NAME,
REQUESTS_RECOVERY);
CREATE INDEX [IDX_QRTZ_FT_G_J]               ON [dbo].[QRTZ_FIRED_TRIGGERS](SCHED_NAME, JOB_GROUP,
JOB_NAME);
CREATE INDEX [IDX_QRTZ_FT_G_T]               ON [dbo].[QRTZ_FIRED_TRIGGERS](SCHED_NAME, TRIGGER_GROUP,
TRIGGER_NAME);
GO

---version-14.0.6012---

PRINT N'Add table [TaskDependencies]';
GO

IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = 'TaskDependencies')
BEGIN
    CREATE TABLE [dbo].[TaskDependencies] (
        [Id] [uniqueidentifier] NOT NULL,
               [TenantId] [uniqueidentifier] NOT NULL,
        [TaskId] [uniqueidentifier] NOT NULL,
        [DependencyTaskId] [uniqueidentifier] NOT NULL,
        [WaitingForDependencies] [bit] NOT NULL,
        [DependencyType] [nvarchar](255),

        CONSTRAINT [PK_TaskDependencies] PRIMARY KEY ([Id]),
        CONSTRAINT [FK_Task_TaskDependencies] FOREIGN KEY ([TaskId]) REFERENCES [Task] ([Id]) ON DELETE NO
ACTION,
        CONSTRAINT [FK_DependencyTask_TaskDependencies] FOREIGN KEY ([DependencyTaskId]) REFERENCES [Task]
([Id]) ON DELETE NO ACTION,
        CONSTRAINT [FK_Tenant_TaskDependencies] FOREIGN KEY ([TenantId]) REFERENCES [Tenant] ([Id]) ON DELETE
NO ACTION
    );
END
GO
```

```
380
381
382 PRINT N'Add column [UseDependencies] to [Task] table';
383 GO
384 IF NOT EXISTS (
385   SELECT
386     *
387   FROM
388     INFORMATION_SCHEMA.COLUMNS
389   WHERE
390     TABLE_NAME = 'Task' AND COLUMN_NAME = 'UseDependencies')
391 BEGIN
392     ALTER TABLE [Task] ADD UseDependencies tinyint NULL;
393 END;
394
395 PRINT N'Add column [DependencyGroupId] to [TaskHistory] table';
396 GO
397 IF NOT EXISTS (
398   SELECT
399     *
400   FROM
401     INFORMATION_SCHEMA.COLUMNS
402   WHERE
403     TABLE_NAME = 'TaskHistory' AND COLUMN_NAME = 'DependencyGroupId')
404 BEGIN
405     ALTER TABLE [TaskHistory] ADD DependencyGroupId UNIQUEIDENTIFIER NULL;
406 END;
407
```