

Known issues

This article summarizes known issues and workarounds for reported issues.

- 1 [Some tasks are incompatible with MariaDB](#)
 - 2 [Data Collection → Inventory → RayVentory dashboard have wrong sorting](#)

 - 3 [Data Analysis → IT Visibility tasks are not performing properly](#)
 - 4 [Missing table error for EOL/EOS tasks](#)
 - 5 ["GuestOS column does not exist" error in Hypervisor Dashboard Overview](#)
 - 6 [Raynet One MSSQL / MariaDB direct tasks throw "Sequence contains no elements" error](#)
 - 7 [Error "SQLite Error 1" when executing the transformation task](#)
 - 8 [Default database is not created \(Docker + MSSQL\)](#)
 - 9 [Shared links not working after migration](#)
 - 10 [Searching for tasks only applies to the current folder](#)
 - 11 [Slow performance of AI-assistant](#)
 - 12 [Citrix Director Devices on MariaDB](#)
 - 13 [Upgrading from 14.0 Update 1 to Data Hub 14.1 could fail under certain conditions and need manual execution of SQL script](#)
-

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:

```
1 SELECT COUNT(*) INTO @index_exists
2 FROM information_schema.statistics
3 WHERE table_schema = DATABASE()
4 AND table_name = 'DataTransformation-result_devices'
5 AND index_name = 'device_import_id_idx';
6
7 -- Create index only if it does not exist
8 IF @index_exists = 0 THEN
9     CREATE INDEX device_import_id_idx ON `DataTransformation-result_devices` (import_id);
10 END IF;
11
12 SELECT COUNT(*) INTO @index_exists
13 FROM information_schema.statistics
14 WHERE table_schema = DATABASE()
15 AND table_name = 'DataTransformation-result_software_summary'
16 AND index_name = 'software_id_idx';
17
18 -- Create index only if it does not exist
19 IF @index_exists = 0 THEN
20     CREATE INDEX software_id_idx ON `DataTransformation-result_software_summary` (Id);
21 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:

```
1 SELECT
2     COUNT(DISTINCT `dtrd`.`device_name`) AS `device_name_Count`,
3     `cs`.`vendor` as Vendor,
4     `cs`.`name` as Name,
5     `cs`.`version` as Version,
6     `csf`.`title` AS `Functionality`,
7     `cs`.`EndOfLifeDate`,
8     `cs`.`Support`,
9     `cs`.`ExtendedSupportDate`
10 FROM `Catalog_v2-SoftwareOnDeviceMatching` `csm`
11 INNER JOIN `DataTransformation-result_devices` `dtrd` ON `dtrd`.`import_id` = `csm`.`computerid`
12 INNER JOIN `Catalog_v2-Software` `cs` ON `cs`.`softwareid` = `csm`.`softwareid`
13 LEFT JOIN `Catalog_v2-SoftwareFunctionMatching` `csfm` ON `csfm`.`softwareid` = `cs`.`softwareid`
```

```
14 LEFT JOIN `Catalog_v2-SoftwareFunction` `csf` ON `csf`.`functionid` = `csfm`.`functionid`
15 WHERE `name` IS NOT NULL
16 GROUP BY
17   `cs`.`vendor`,
18   `cs`.`name`,
19   `cs`.`version`,
20   `csf`.`title`,
21   `cs`.`endoflifedate`,
22   `cs`.`support`,
23   `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:

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

with this:

```
1 `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:

```
1 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:

```
1 SELECT DISTINCT
2   LEFT(REPLACE([Name], ' ', '_'), 255) AS import_id,
3   'RayVentory' AS import_data_source_id,
4   LEFT(REPLACE([Name], ' ', '_'), 255) AS device_key,
5   'unknown_costcenter' AS import_org_level_2_id,
6   LEFT(REPLACE([Name], ' ', '_'), 255) AS device_name,
7   NULL AS device_manufacturer,
8   NULL AS device_model,
9   NULL AS serial_number,
10  NULL AS import_operating_system_id,
11  'xen_cluster' AS import_device_type_id,
12  'Active' AS import_device_status_id,
13  NULL AS import_domain_id,
14  NULL AS import_user_id,
15  NULL AS ip_address,
16  NULL AS mac_address,
17  LEFT(REPLACE([Name], ' ', '_'), 255) AS fqdn,
18  NULL AS inventory_date,
19  NULL AS installation_date,
20  NULL AS cpu_chip_count,
21  NULL AS cpu_core_count,
22  NULL AS cpu_speed,
23  NULL AS import_cpu_type_id,
24  NULL AS ram,
25  NULL AS storage,
26  NULL AS graphics,
27  NULL AS network,
28  NULL AS bios,
29  NULL AS cpu_core_count_limited
30 FROM MonitorData.Hypervisor
31
32 UNION ALL
33
34 SELECT DISTINCT
35   LEFT(DnsName, 255) AS import_id,
36   'RayVentory' AS import_data_source_id,
37   LEFT(DnsName, 255) AS device_key,
38   'unknown_costcenter' AS import_org_level_2_id,
39   LEFT(HostedMachineName, 255) AS device_name,
40   'Citrix Systems, Inc.' AS device_manufacturer,
41   'Citrix XenDesktop' AS device_model,
42   [Hash] AS serial_number,
43   OStype AS import_operating_system_id,
44   'virtual_server' AS import_device_type_id,
45   'Active' AS import_device_status_id,
```

```

46 LEFT(SUBSTRING(DnsName, CHARINDEX('.',DnsName,0)+1, LEN(DnsName)-CHARINDEX('.',DnsName,0)), 255) AS
import_domain_id,
47 NULL AS import_user_id,
48 NULL AS ip_address,
49 NULL AS mac_address,
50 LEFT(DnsName, 255) AS fqdn,
51 NULL AS inventory_date,
52 NULL AS installation_date,
53 NULL AS cpu_chip_count,
54 NULL AS cpu_core_count,
55 NULL AS cpu_speed,
56 NULL AS import_cpu_type_id,
57 NULL AS ram,
58 NULL AS storage,
59 NULL AS graphics,
60 NULL AS network,
61 NULL AS bios,
62 NULL AS cpu_core_count_limited
63 FROM MonitorData.Machine
64 WHERE ISNULL(DnsName, '') <> ''
65
66 UNION ALL
67
68 SELECT DISTINCT
69 LEFT(HostingServerName, 255) AS import_id,
70 'RayVentory' AS import_data_source_id,
71 LEFT(HostingServerName, 255) AS device_key,
72 'unknown_costcenter' AS import_org_level_2_id,
73 LEFT(HostingServerName, 255) AS device_name,
74 NULL AS device_manufacturer,
75 NULL AS device_model,
76 NULL AS serial_number,
77 NULL AS import_operating_system_id,
78 'xen_host' AS import_device_type_id,
79 'Active' AS import_device_status_id,
80 LEFT(SUBSTRING(HostingServerName, CHARINDEX('.',HostingServerName,0)+1, LEN(HostingServerName)-
CHARINDEX('.',HostingServerName,0)), 255) AS import_domain_id,
81 NULL AS import_user_id,
82 NULL AS ip_address,
83 NULL AS mac_address,
84 LEFT(HostingServerName, 255) AS fqdn,
85 NULL AS inventory_date,
86 NULL AS installation_date,
87 NULL AS cpu_chip_count,
88 NULL AS cpu_core_count,
89 NULL AS cpu_speed,
90 NULL AS import_cpu_type_id,
91 NULL AS ram,
92 NULL AS storage,
93 NULL AS graphics,
94 NULL AS network,
95 NULL AS bios,
96 NULL AS cpu_core_count_limited
97 FROM MonitorData.Machine
98 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:

```
1 PRINT N'Add quartz tables';
2 DROP INDEX IF EXISTS [IDX_QRTZ_T_J] ON [dbo].[QRTZ_TRIGGERS];
3 DROP INDEX IF EXISTS [IDX_QRTZ_T_JG] ON [dbo].[QRTZ_TRIGGERS];
4 DROP INDEX IF EXISTS [IDX_QRTZ_T_C] ON [dbo].[QRTZ_TRIGGERS];
5 DROP INDEX IF EXISTS [IDX_QRTZ_T_G] ON [dbo].[QRTZ_TRIGGERS];
6 DROP INDEX IF EXISTS [IDX_QRTZ_T_G_J] ON [dbo].[QRTZ_TRIGGERS];
7 DROP INDEX IF EXISTS [IDX_QRTZ_T_STATE] ON [dbo].[QRTZ_TRIGGERS];
8 DROP INDEX IF EXISTS [IDX_QRTZ_T_N_STATE] ON [dbo].[QRTZ_TRIGGERS];
9 DROP INDEX IF EXISTS [IDX_QRTZ_T_N_G_STATE] ON [dbo].[QRTZ_TRIGGERS];
10 DROP INDEX IF EXISTS [IDX_QRTZ_T_NEXT_FIRE_TIME] ON [dbo].[QRTZ_TRIGGERS];
11 DROP INDEX IF EXISTS [IDX_QRTZ_T_NFT_ST] ON [dbo].[QRTZ_TRIGGERS];
12 DROP INDEX IF EXISTS [IDX_QRTZ_T_NFT_MISFIRE] ON [dbo].[QRTZ_TRIGGERS];
13 DROP INDEX IF EXISTS [IDX_QRTZ_T_NFT_ST_MISFIRE] ON [dbo].[QRTZ_TRIGGERS];
14 DROP INDEX IF EXISTS [IDX_QRTZ_T_NFT_ST_MISFIRE_GRP] ON [dbo].[QRTZ_TRIGGERS];
15 DROP INDEX IF EXISTS [IDX_QRTZ_FT_TRIG_INST_NAME] ON [dbo].[QRTZ_FIRED_TRIGGERS];
16 DROP INDEX IF EXISTS [IDX_QRTZ_FT_INST_JOB_REQ_RCVRY] ON [dbo].[QRTZ_FIRED_TRIGGERS];
17 DROP INDEX IF EXISTS [IDX_QRTZ_FT_J_G] ON [dbo].[QRTZ_FIRED_TRIGGERS];
18 DROP INDEX IF EXISTS [IDX_QRTZ_FT_JG] ON [dbo].[QRTZ_FIRED_TRIGGERS];
19 DROP INDEX IF EXISTS [IDX_QRTZ_FT_T_G] ON [dbo].[QRTZ_FIRED_TRIGGERS];
20 DROP INDEX IF EXISTS [IDX_QRTZ_FT_TG] ON [dbo].[QRTZ_FIRED_TRIGGERS];
21 DROP INDEX IF EXISTS [IDX_QRTZ_FT_G_J] ON [dbo].[QRTZ_FIRED_TRIGGERS];
22 DROP INDEX IF EXISTS [IDX_QRTZ_FT_G_T] ON [dbo].[QRTZ_FIRED_TRIGGERS];
23
24 IF OBJECT_ID(N'[dbo].[FK_QRTZ_TRIGGERS_QRTZ_JOB_DETAILS]', N'F') IS NOT NULL
25 ALTER TABLE [dbo].[QRTZ_TRIGGERS] DROP CONSTRAINT [FK_QRTZ_TRIGGERS_QRTZ_JOB_DETAILS];
26
27 IF OBJECT_ID(N'[dbo].[FK_QRTZ_CRON_TRIGGERS_QRTZ_TRIGGERS]', N'F') IS NOT NULL
28 ALTER TABLE [dbo].[QRTZ_CRON_TRIGGERS] DROP CONSTRAINT [FK_QRTZ_CRON_TRIGGERS_QRTZ_TRIGGERS];
29
30 IF OBJECT_ID(N'[dbo].[FK_QRTZ_SIMPLE_TRIGGERS_QRTZ_TRIGGERS]', N'F') IS NOT NULL
31 ALTER TABLE [dbo].[QRTZ_SIMPLE_TRIGGERS] DROP CONSTRAINT [FK_QRTZ_SIMPLE_TRIGGERS_QRTZ_TRIGGERS];
32
33 IF OBJECT_ID(N'[dbo].[FK_QRTZ_SIMPROP_TRIGGERS_QRTZ_TRIGGERS]', N'F') IS NOT NULL
34 ALTER TABLE [dbo].[QRTZ_SIMPROP_TRIGGERS] DROP CONSTRAINT [FK_QRTZ_SIMPROP_TRIGGERS_QRTZ_TRIGGERS];
35
36 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]'))
37 ALTER TABLE [dbo].[QRTZ_JOB_LISTENERS] DROP CONSTRAINT [FK_QRTZ_JOB_LISTENERS_QRTZ_JOB_DETAILS];
38
39 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]'))
40 ALTER TABLE [dbo].[QRTZ_TRIGGER_LISTENERS] DROP CONSTRAINT [FK_QRTZ_TRIGGER_LISTENERS_QRTZ_TRIGGERS];
41
42 IF OBJECT_ID(N'[dbo].[QRTZ_CALENDARS]', N'U') IS NOT NULL
```

```

43 DROP TABLE [dbo].[QRTZ_CALENDARS];
44
45 IF OBJECT_ID(N'[dbo].[QRTZ_CRON_TRIGGERS]', N'U') IS NOT NULL
46 DROP TABLE [dbo].[QRTZ_CRON_TRIGGERS];
47
48 IF OBJECT_ID(N'[dbo].[QRTZ_BLOB_TRIGGERS]', N'U') IS NOT NULL
49 DROP TABLE [dbo].[QRTZ_BLOB_TRIGGERS];
50
51 IF OBJECT_ID(N'[dbo].[QRTZ_FIRED_TRIGGERS]', N'U') IS NOT NULL
52 DROP TABLE [dbo].[QRTZ_FIRED_TRIGGERS];
53
54 IF OBJECT_ID(N'[dbo].[QRTZ_PAUSED_TRIGGER_GRPS]', N'U') IS NOT NULL
55 DROP TABLE [dbo].[QRTZ_PAUSED_TRIGGER_GRPS];
56
57 IF OBJECT_ID(N'[dbo].[QRTZ_JOB_LISTENERS]', N'U') IS NOT NULL
58 DROP TABLE [dbo].[QRTZ_JOB_LISTENERS];
59
60 IF OBJECT_ID(N'[dbo].[QRTZ_SCHEDULER_STATE]', N'U') IS NOT NULL
61 DROP TABLE [dbo].[QRTZ_SCHEDULER_STATE];
62
63 IF OBJECT_ID(N'[dbo].[QRTZ_LOCKS]', N'U') IS NOT NULL
64 DROP TABLE [dbo].[QRTZ_LOCKS];
65
66 IF OBJECT_ID(N'[dbo].[QRTZ_TRIGGER_LISTENERS]', N'U') IS NOT NULL
67 DROP TABLE [dbo].[QRTZ_TRIGGER_LISTENERS];
68
69 IF OBJECT_ID(N'[dbo].[QRTZ_JOB_DETAILS]', N'U') IS NOT NULL
70 DROP TABLE [dbo].[QRTZ_JOB_DETAILS];
71
72 IF OBJECT_ID(N'[dbo].[QRTZ_SIMPLE_TRIGGERS]', N'U') IS NOT NULL
73 DROP TABLE [dbo].[QRTZ_SIMPLE_TRIGGERS];
74
75 IF OBJECT_ID(N'[dbo].[QRTZ_SIMPROP_TRIGGERS]', N'U') IS NOT NULL
76 DROP TABLE [dbo].[QRTZ_SIMPROP_TRIGGERS];
77
78 IF OBJECT_ID(N'[dbo].[QRTZ_TRIGGERS]', N'U') IS NOT NULL
79 DROP TABLE [dbo].[QRTZ_TRIGGERS];
80
81 CREATE TABLE [dbo].[QRTZ_CALENDARS] (
82     [SCHED_NAME] nvarchar(120) NOT NULL,
83     [CALENDAR_NAME] nvarchar(200) NOT NULL,
84     [CALENDAR] varbinary(max) NOT NULL
85 );
86
87 CREATE TABLE [dbo].[QRTZ_CRON_TRIGGERS] (
88     [SCHED_NAME] nvarchar(120) NOT NULL,
89     [TRIGGER_NAME] nvarchar(150) NOT NULL,
90     [TRIGGER_GROUP] nvarchar(150) NOT NULL,
91     [CRON_EXPRESSION] nvarchar(120) NOT NULL,
92     [TIME_ZONE_ID] nvarchar(80)
93 );
94
95 CREATE TABLE [dbo].[QRTZ_FIRED_TRIGGERS] (
96     [SCHED_NAME] nvarchar(120) NOT NULL,
97     [ENTRY_ID] nvarchar(140) NOT NULL,
98     [TRIGGER_NAME] nvarchar(150) NOT NULL,
99     [TRIGGER_GROUP] nvarchar(150) NOT NULL,
100    [INSTANCE_NAME] nvarchar(200) NOT NULL,

```



```

101     [FIRED_TIME] bigint NOT NULL,
102     [SCHED_TIME] bigint NOT NULL,
103     [PRIORITY] int NOT NULL,
104     [STATE] nvarchar(16) NOT NULL,
105     [JOB_NAME] nvarchar(150) NULL,
106     [JOB_GROUP] nvarchar(150) NULL,
107     [IS_NONCONCURRENT] bit NULL,
108     [REQUESTS_RECOVERY] bit NULL
109 );
110
111 CREATE TABLE [dbo].[QRTZ_PAUSED_TRIGGER_GRPS] (
112     [SCHED_NAME] nvarchar(120) NOT NULL,
113     [TRIGGER_GROUP] nvarchar(150) NOT NULL
114 );
115
116 CREATE TABLE [dbo].[QRTZ_SCHEDULER_STATE] (
117     [SCHED_NAME] nvarchar(120) NOT NULL,
118     [INSTANCE_NAME] nvarchar(200) NOT NULL,
119     [LAST_CHECKIN_TIME] bigint NOT NULL,
120     [CHECKIN_INTERVAL] bigint NOT NULL
121 );
122
123 CREATE TABLE [dbo].[QRTZ_LOCKS] (
124     [SCHED_NAME] nvarchar(120) NOT NULL,
125     [LOCK_NAME] nvarchar(40) NOT NULL
126 );
127
128 CREATE TABLE [dbo].[QRTZ_JOB_DETAILS] (
129     [SCHED_NAME] nvarchar(120) NOT NULL,
130     [JOB_NAME] nvarchar(150) NOT NULL,
131     [JOB_GROUP] nvarchar(150) NOT NULL,
132     [DESCRIPTION] nvarchar(250) NULL,
133     [JOB_CLASS_NAME] nvarchar(250) NOT NULL,
134     [IS_DURABLE] bit NOT NULL,
135     [IS_NONCONCURRENT] bit NOT NULL,
136     [IS_UPDATE_DATA] bit NOT NULL,
137     [REQUESTS_RECOVERY] bit NOT NULL,
138     [JOB_DATA] varbinary(max) NULL
139 );
140
141 CREATE TABLE [dbo].[QRTZ_SIMPLE_TRIGGERS] (
142     [SCHED_NAME] nvarchar(120) NOT NULL,
143     [TRIGGER_NAME] nvarchar(150) NOT NULL,
144     [TRIGGER_GROUP] nvarchar(150) NOT NULL,
145     [REPEAT_COUNT] int NOT NULL,
146     [REPEAT_INTERVAL] bigint NOT NULL,
147     [TIMES_TRIGGERED] int NOT NULL
148 );
149
150 CREATE TABLE [dbo].[QRTZ_SIMPROP_TRIGGERS] (
151     [SCHED_NAME] nvarchar(120) NOT NULL,
152     [TRIGGER_NAME] nvarchar(150) NOT NULL,
153     [TRIGGER_GROUP] nvarchar(150) NOT NULL,
154     [STR_PROP_1] nvarchar(512) NULL,
155     [STR_PROP_2] nvarchar(512) NULL,
156     [STR_PROP_3] nvarchar(512) NULL,
157     [INT_PROP_1] int NULL,
158     [INT_PROP_2] int NULL,

```

```

159     [LONG_PROP_1] bigint NULL,
160     [LONG_PROP_2] bigint NULL,
161     [DEC_PROP_1] numeric(13,4) NULL,
162     [DEC_PROP_2] numeric(13,4) NULL,
163     [BOOL_PROP_1] bit NULL,
164     [BOOL_PROP_2] bit NULL,
165     [TIME_ZONE_ID] nvarchar(80) NULL
166 );
167
168 CREATE TABLE [dbo].[QRTZ_BLOB_TRIGGERS] (
169     [SCHED_NAME] nvarchar(120) NOT NULL,
170     [TRIGGER_NAME] nvarchar(150) NOT NULL,
171     [TRIGGER_GROUP] nvarchar(150) NOT NULL,
172     [BLOB_DATA] varbinary(max) NULL
173 );
174
175 CREATE TABLE [dbo].[QRTZ_TRIGGERS] (
176     [SCHED_NAME] nvarchar(120) NOT NULL,
177     [TRIGGER_NAME] nvarchar(150) NOT NULL,
178     [TRIGGER_GROUP] nvarchar(150) NOT NULL,
179     [JOB_NAME] nvarchar(150) NOT NULL,
180     [JOB_GROUP] nvarchar(150) NOT NULL,
181     [DESCRIPTION] nvarchar(250) NULL,
182     [NEXT_FIRE_TIME] bigint NULL,
183     [PREV_FIRE_TIME] bigint NULL,
184     [PRIORITY] int NULL,
185     [TRIGGER_STATE] nvarchar(16) NOT NULL,
186     [TRIGGER_TYPE] nvarchar(8) NOT NULL,
187     [START_TIME] bigint NOT NULL,
188     [END_TIME] bigint NULL,
189     [CALENDAR_NAME] nvarchar(200) NULL,
190     [MISFIRE_INSTR] int NULL,
191     [JOB_DATA] varbinary(max) NULL
192 );
193 GO
194
195 ALTER TABLE [dbo].[QRTZ_CALENDARS] WITH NOCHECK ADD
196     CONSTRAINT [PK_QRTZ_CALENDARS] PRIMARY KEY CLUSTERED
197     (
198         [SCHED_NAME],
199         [CALENDAR_NAME]
200     );
201 GO
202
203 ALTER TABLE [dbo].[QRTZ_CRON_TRIGGERS] WITH NOCHECK ADD
204     CONSTRAINT [PK_QRTZ_CRON_TRIGGERS] PRIMARY KEY CLUSTERED
205     (
206         [SCHED_NAME],
207         [TRIGGER_NAME],
208         [TRIGGER_GROUP]
209     );
210 GO
211
212 ALTER TABLE [dbo].[QRTZ_FIRED_TRIGGERS] WITH NOCHECK ADD
213     CONSTRAINT [PK_QRTZ_FIRED_TRIGGERS] PRIMARY KEY CLUSTERED
214     (
215         [SCHED_NAME],
216         [ENTRY_ID]

```

```

217 );
218 GO
219
220 ALTER TABLE [dbo].[QRTZ_PAUSED_TRIGGER_GRPS] WITH NOCHECK ADD
221     CONSTRAINT [PK_QRTZ_PAUSED_TRIGGER_GRPS] PRIMARY KEY CLUSTERED
222     (
223         [SCHED_NAME],
224         [TRIGGER_GROUP]
225     );
226 GO
227
228 ALTER TABLE [dbo].[QRTZ_SCHEDULER_STATE] WITH NOCHECK ADD
229     CONSTRAINT [PK_QRTZ_SCHEDULER_STATE] PRIMARY KEY CLUSTERED
230     (
231         [SCHED_NAME],
232         [INSTANCE_NAME]
233     );
234 GO
235
236 ALTER TABLE [dbo].[QRTZ_LOCKS] WITH NOCHECK ADD
237     CONSTRAINT [PK_QRTZ_LOCKS] PRIMARY KEY CLUSTERED
238     (
239         [SCHED_NAME],
240         [LOCK_NAME]
241     );
242 GO
243
244 ALTER TABLE [dbo].[QRTZ_JOB_DETAILS] WITH NOCHECK ADD
245     CONSTRAINT [PK_QRTZ_JOB_DETAILS] PRIMARY KEY CLUSTERED
246     (
247         [SCHED_NAME],
248         [JOB_NAME],
249         [JOB_GROUP]
250     );
251 GO
252
253 ALTER TABLE [dbo].[QRTZ_SIMPLE_TRIGGERS] WITH NOCHECK ADD
254     CONSTRAINT [PK_QRTZ_SIMPLE_TRIGGERS] PRIMARY KEY CLUSTERED
255     (
256         [SCHED_NAME],
257         [TRIGGER_NAME],
258         [TRIGGER_GROUP]
259     );
260 GO
261
262 ALTER TABLE [dbo].[QRTZ_SIMPROP_TRIGGERS] WITH NOCHECK ADD
263     CONSTRAINT [PK_QRTZ_SIMPROP_TRIGGERS] PRIMARY KEY CLUSTERED
264     (
265         [SCHED_NAME],
266         [TRIGGER_NAME],
267         [TRIGGER_GROUP]
268     );
269 GO
270
271 ALTER TABLE [dbo].[QRTZ_TRIGGERS] WITH NOCHECK ADD
272     CONSTRAINT [PK_QRTZ_TRIGGERS] PRIMARY KEY CLUSTERED
273     (
274         [SCHED_NAME],

```

```

275     [TRIGGER_NAME],
276     [TRIGGER_GROUP]
277 );
278 GO
279
280 ALTER TABLE [dbo].[QRTZ_BLOB_TRIGGERS] WITH NOCHECK ADD
281     CONSTRAINT [PK_QRTZ_BLOB_TRIGGERS] PRIMARY KEY CLUSTERED
282     (
283         [SCHED_NAME],
284         [TRIGGER_NAME],
285         [TRIGGER_GROUP]
286     );
287 GO
288
289 ALTER TABLE [dbo].[QRTZ_CRON_TRIGGERS] ADD
290     CONSTRAINT [FK_QRTZ_CRON_TRIGGERS_QRTZ_TRIGGERS] FOREIGN KEY
291     (
292         [SCHED_NAME],
293         [TRIGGER_NAME],
294         [TRIGGER_GROUP]
295     ) REFERENCES [dbo].[QRTZ_TRIGGERS] (
296         [SCHED_NAME],
297         [TRIGGER_NAME],
298         [TRIGGER_GROUP]
299     ) ON DELETE CASCADE;
300 GO
301
302 ALTER TABLE [dbo].[QRTZ_SIMPLE_TRIGGERS] ADD
303     CONSTRAINT [FK_QRTZ_SIMPLE_TRIGGERS_QRTZ_TRIGGERS] FOREIGN KEY
304     (
305         [SCHED_NAME],
306         [TRIGGER_NAME],
307         [TRIGGER_GROUP]
308     ) REFERENCES [dbo].[QRTZ_TRIGGERS] (
309         [SCHED_NAME],
310         [TRIGGER_NAME],
311         [TRIGGER_GROUP]
312     ) ON DELETE CASCADE;
313 GO
314
315 ALTER TABLE [dbo].[QRTZ_SIMPROP_TRIGGERS] ADD
316     CONSTRAINT [FK_QRTZ_SIMPROP_TRIGGERS_QRTZ_TRIGGERS] FOREIGN KEY
317     (
318         [SCHED_NAME],
319         [TRIGGER_NAME],
320         [TRIGGER_GROUP]
321     ) REFERENCES [dbo].[QRTZ_TRIGGERS] (
322         [SCHED_NAME],
323         [TRIGGER_NAME],
324         [TRIGGER_GROUP]
325     ) ON DELETE CASCADE;
326 GO
327
328 ALTER TABLE [dbo].[QRTZ_TRIGGERS] ADD
329     CONSTRAINT [FK_QRTZ_TRIGGERS_QRTZ_JOB_DETAILS] FOREIGN KEY
330     (
331         [SCHED_NAME],
332         [JOB_NAME],

```

```

333     [JOB_GROUP]
334 ) REFERENCES [dbo].[QRTZ_JOB_DETAILS] (
335     [SCHED_NAME],
336     [JOB_NAME],
337     [JOB_GROUP]
338 );
339 GO
340
341 -- Create indexes
342 CREATE INDEX [IDX_QRTZ_T_G_J] ON [dbo].[QRTZ_TRIGGERS] (SCHED_NAME, JOB_GROUP, JOB_NAME);
343 CREATE INDEX [IDX_QRTZ_T_C] ON [dbo].[QRTZ_TRIGGERS] (SCHED_NAME, CALENDAR_NAME);
344
345 CREATE INDEX [IDX_QRTZ_T_N_G_STATE] ON [dbo].[QRTZ_TRIGGERS] (SCHED_NAME, TRIGGER_GROUP,
TRIGGER_STATE);
346 CREATE INDEX [IDX_QRTZ_T_STATE] ON [dbo].[QRTZ_TRIGGERS] (SCHED_NAME, TRIGGER_STATE);
347 CREATE INDEX [IDX_QRTZ_T_N_STATE] ON [dbo].[QRTZ_TRIGGERS] (SCHED_NAME, TRIGGER_NAME,
TRIGGER_GROUP, TRIGGER_STATE);
348 CREATE INDEX [IDX_QRTZ_T_NEXT_FIRE_TIME] ON [dbo].[QRTZ_TRIGGERS] (SCHED_NAME, NEXT_FIRE_TIME);
349 CREATE INDEX [IDX_QRTZ_T_NFT_ST] ON [dbo].[QRTZ_TRIGGERS] (SCHED_NAME, TRIGGER_STATE,
NEXT_FIRE_TIME);
350 CREATE INDEX [IDX_QRTZ_T_NFT_ST_MISFIRE] ON [dbo].[QRTZ_TRIGGERS] (SCHED_NAME, MISFIRE_INSTR,
NEXT_FIRE_TIME, TRIGGER_STATE);
351 CREATE INDEX [IDX_QRTZ_T_NFT_ST_MISFIRE_GRP] ON [dbo].[QRTZ_TRIGGERS] (SCHED_NAME, MISFIRE_INSTR,
NEXT_FIRE_TIME, TRIGGER_GROUP, TRIGGER_STATE);
352
353 CREATE INDEX [IDX_QRTZ_FT_INST_JOB_REQ_RCVRY] ON [dbo].[QRTZ_FIRED_TRIGGERS] (SCHED_NAME, INSTANCE_NAME,
REQUESTS_RECOVERY);
354 CREATE INDEX [IDX_QRTZ_FT_G_J] ON [dbo].[QRTZ_FIRED_TRIGGERS] (SCHED_NAME, JOB_GROUP,
JOB_NAME);
355 CREATE INDEX [IDX_QRTZ_FT_G_T] ON [dbo].[QRTZ_FIRED_TRIGGERS] (SCHED_NAME, TRIGGER_GROUP,
TRIGGER_NAME);
356 GO
357
358 ---version-14.0.6012---
359
360 PRINT N'Add table [TaskDependencies]';
361 GO
362
363 IF NOT EXISTS (SELECT 1 FROM sys.tables WHERE name = 'TaskDependencies')
364 BEGIN
365     CREATE TABLE [dbo].[TaskDependencies] (
366         [Id] [uniqueidentifier] NOT NULL,
367         [TenantId] [uniqueidentifier] NOT NULL,
368         [TaskId] [uniqueidentifier] NOT NULL,
369         [DependencyTaskId] [uniqueidentifier] NOT NULL,
370         [WaitingForDependencies] [bit] NOT NULL,
371         [DependencyType] [nvarchar](255),
372
373         CONSTRAINT [PK_TaskDependencies] PRIMARY KEY ([Id]),
374         CONSTRAINT [FK_Task_TaskDependencies] FOREIGN KEY ([TaskId]) REFERENCES [Task] ([Id]) ON DELETE NO
ACTION,
375         CONSTRAINT [FK_DependencyTask_TaskDependencies] FOREIGN KEY ([DependencyTaskId]) REFERENCES [Task]
([Id]) ON DELETE NO ACTION,
376         CONSTRAINT [FK_Tenant_TaskDependencies] FOREIGN KEY ([TenantId]) REFERENCES [Tenant] ([Id]) ON DELETE
NO ACTION
377     );
378 END
379 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
```