CDC Sample script for testing and troubleshooting

Last updated by | Vitor Tomaz | Jun 8, 2022 at 5:36 AM PDT

Contents

Sample script for enabling CDC, checking metadata, and viewing captured changes

Issue

This sample script will help you getting familiar with the Change Data Capture (CDC) feature. The goal is to enable you to answer customer questions through steps like:

- quickly create a test environment on Managed Instance
- review the DMVs and system tables associated with CDC
- see what CDC jobs are created and how they are scheduled and configured
- perform a data change and see how it is tracked in the DMVs and system tables
- perform a schema change on the data table and how it is handled by CDC
- disable and remove the CDC environment

Additional content

CDC Change Tables cleanup slow or not working (shows how the DMV queries from this sample script can help with troubleshooting)

CDC Kusto Queries for troubleshooting (run these queries against your test MI to get an idea on available telemetry)

The TSGs in the Change Data Capture - CDC on Azure SQL Database section might provide additional guidance.

Step-by-Step commands explained

Create database and enable CDC

```
/** Start of Script **/

-- create test database
CREATE DATABASE CDC_Publisher
GO

USE CDC_Publisher
-- enable database for CDC
if not exists (select * from master.sys.databases where name = 'CDC_Publisher' and is_cdc_enabled = 1)
        EXEC sys.sp_cdc_enable_db
go

-- create a table and add some data
USE CDC_Publisher
CREATE TABLE t1 (ID INT PRIMARY KEY NOT NULL, c1 VARCHAR(100))
GO
INSERT INTO t1 (ID, c1) VALUES (1, 'row one')
INSERT INTO t1 (ID, c1) VALUES (2, 'row two')
GO

-- enable table for CDC
EXEC sys.sp_cdc_enable_table  @source_schema = 'dbo', @source_name = 't1', @role_name = 'cdc_Admin'
GO
/*
Job 'cdc.CDC_Publisher_capture' started successfully.
Job 'cdc.CDC_Publisher_cleanup' started successfully.
*/
```

sys.sp_cdc_enable_db - creates the CDC metadata tables for the database
sys.sp_cdc_enable_table - creates a tracking table, a capture instance and initial metadata for the tracked table

After enabling the first table for CDC, the capture job and the cleanup job are created. The capture job harvests changes for all capture instances from the transaction log, whereas the cleanup job trims outdated metadata from the system.

DMVs related to CDC

```
-- take a look at system tables and views:
select * from CDC.change_tables     -- details on active CDC tables
select * from CDC.captured_columns  -- what columns are tracked for each table
select * from CDC.index_columns     -- PK/unique column of tracked tables
select * from CDC.dbo_t1_ct         -- this is the actual tracking table
select * from CDC.lsn_time_mapping  -- mapping between LSNs and point in time
select * from CDC.ddl_history       -- DDL changes (only filled after capture job has run)
select * from dbo.systranschemas    -- DDL changes
go
-- display summary details through stored proc:
exec sp_cdc_help_change_data_capture @source_schema = 'dbo', @source_name = 't1'
go
```

Please run these queries here after the initial configuration, and also later after changing user data and tables (DML and DDL). The same set of queries can also help you with getting details from a customer environment. You should specifically take note of the `start_lsn` and the various usages of Log Sequence Numbers (LSNs) for managing change tracking, sequence of events, and cleanup.

cdc.change_tables - details on active CDC tables
cdc.captured_columns - what columns are tracked for each table
cdc.index_columns - PK/unique column of tracked tables
cdc.<capture_instance>_CT - this is the actual tracking table for data changes
cdc.lsn_time_mapping - mapping between LSNs and point in time
cdc.ddl_history - DDL changes (only filled after capture job has run)
systranschemas - DDL changes

CDC Job details

```
-- display details on the capture and cleanup job:
exec sys.sp_cdc_help_jobs
select * from msdb..cdc_jobs
select * from msdb..sysjobs where name like 'cdc%'
go
```

The output of `sys.sp_cdc_help_jobs` shows you the configuration details of capture and cleanup. The direct queries on the MSDB tables provide the same details, plus some job-specific information.

```
-- sample output from sp_cdc_help_jobs:
job_id          job_type job_name            maxtrans  maxscans continuous pollinginterval retention threshold
--------------- -------- ------------------- -------- --------- ---------- --------------- --------- ---------
5C7359FB-52...  capture  cdc.dbname_capture  500      10        1          5               0         0
FEDB9B08-F9...  cleanup  cdc.dbname_cleanup  0        0         0          0               4320      5000
```

sys.sp_cdc_help_jobs

The `maxtrans` value for the capture job configures the batch size with which transactions are harvested from the transaction log (default is 500). The other columns show details on the job execution configuration e.g. if it runs on a schedule or continuously.

The retention period of the CDC metadata is set through the cleanup job. Here it is the default of 4320 minutes = 72 hours = 3 days, which you can see on the `retention` column. Also note the value that is returned on the `threshold` column; it represents the Delete Batch Size for the cleanup, with a default value of 5000.

You can use sys.sp_cdc_change_job to modify the configuration values of the CDC cleanup or capture job.

Start capture job manually

```
-- Start capture job through code:
EXEC sp_MScdc_capture_job

-- Call sp_replflush to release the full logreader context:
exec sp_replflush
go
```

You can start the capture job manually by executing the `sp_MScdc_capture_job` stored procedure - this is the same call that the capture job is executing.
The procedure call will not return - the statement in the query window remains in status "Executing query..." and you have to cancel the execution. But note that cancelling the execution in the query window won't stop the actual capture job execution.

Also note that the client that has executed `sp_MScdc_capture_job` has also assumed the "Log Reader" execution context. You have to either close the connection or execute sp_replflush to release the "Log Reader" context.

Instead of executing `sp_MScdc_capture_job`, you can also start and stop the SQL Agent jobs through the following procedure calls:

```
-- start and stop the SQL Agent capture job
exec sys.sp_cdc_start_job @job_type= 'capture'
exec sys.sp_cdc_stop_job @job_type= 'capture'

-- start and stop the SQL Agent cleanup job
exec sys.sp_cdc_start_job @job_type= 'cleanup'
exec sys.sp_cdc_stop_job @job_type= 'cleanup'
```

The SQL Agent jobs give you better control, but might not be available e.g. if the customer has deleted or managed to break them.

If the capture job is already running, you will get the following error message either on the SQL Server Agent job history or as query result:

```
Msg 22903, Level 16, State 1, Procedure sp_repldone, Line 1 [Batch Start Line 47]
Another connection with session ID 97 is already running 'sp_replcmds' for Change Data Capture in the current
Msg 22864, Level 16, State 1, Procedure sp_MScdc_capture_job, Line 102 [Batch Start Line 47]
The call to sp_MScdc_capture_job by the Capture Job for database 'CDC_Publisher' failed. Look at previous erro
```

Note that it is telling you the session ID of the existing capture job. If you need to stop an orphaned job execution, you can kill the session ID ("kill 97").

Apply data change

```
-- Apply some changes:
INSERT INTO t1 (ID, c1) VALUES (3, 'row three')
GO
UPDATE t1 SET c1 = 'update ' + c1 WHERE ID = 2
go

-- in the tracking table:
select * from CDC.dbo_t1_ct
-- will return rows only if the capture job was running!
-- will return 1 row for the insert:  __$operation = 2
-- will return 2 rows for the update: __$operation = 3 for the old data, = 4 for the new data

-- the CDC changes wrapper functions will give you:
declare @from_lsn binary(10), @to_lsn binary(10)
set @from_lsn = sys.fn_cdc_get_min_lsn('dbo_t1')
set @to_lsn = sys.fn_cdc_get_max_lsn()
select * from cdc.fn_cdc_get_all_changes_dbo_t1(@from_lsn, @to_lsn, 'all')
select * from cdc.fn_cdc_get_net_changes_dbo_t1(@from_lsn, @to_lsn, 'all')
```

sys.fn_cdc_get_min_lsn returns the start_lsn column value for the specified capture instance from `cdc.change_tables`. This value represents the low endpoint of the validity interval for the capture instance. Older LSNs have already been removed by the cleanup job. Instead of the min_lsn, you can also specify a higher custom LSN as a starting point, e.g. the low watermark from a recent sync operation that has already retrieved some of the changes.

sys.fn_cdc_get_max_lsn returns the maximum log sequence number (LSN) from the start_lsn column in the cdc.lsn_time_mapping system table; you can use this function to return the high endpoint of the change data capture timeline for any capture instance.

cdc.fn_cdc_get_all_changes_<capture_instance> returns one row for each change applied to the source table within the specified log sequence number (LSN) range.

cdc.fn_cdc_get_net_changes_<capture_instance> returns one net change row for each source row changed within the specified Log Sequence Numbers (LSN) range; several sequential changes are rolled up into one net change.

Correlate entry time and LSN

```
-- correlate LSN with a point in time (uses CDC.lsn_time_mapping internally)
declare @pit_lsn binary(10)
declare @lsn_pit datetime
-- get closest LSN for a datetime
select @pit_lsn = sys.fn_cdc_map_time_to_lsn('largest less than or equal', GETDATE())
select @pit_lsn, GETDATE()
-- get closest datetime for an LSN
select @lsn_pit = sys.fn_cdc_map_lsn_to_time(@pit_lsn)
select @pit_lsn as 'LSN from initial datetime', @lsn_pit as 'Datetime from LSN', GETDATE() as 'Initial datetim
go

-- sample output:
LSN from Initial datetime  Datetime from LSN       Initial datetime
-------------------------- ----------------------- -----------------------
0x00000037000001F00001     2022-04-27 12:40:49.443 2022-04-27 12:41:12.060
```

sys.fn_cdc_map_time_to_lsn returns the LSN value from the start_lsn column in the cdc.lsn_time_mapping system table for the specified time, or its nearest match. You can use this function to systematically map datetime ranges into LSN-based ranges which are needed by the CDC enumeration functions.

sys.fn_cdc_map_lsn_to_time returns the datetime value from the tran_end_time column in the cdc.lsn_time_mapping system table for the specified LSN. You can use this function to systematically map LSN ranges to datetime ranges in a change table.

Retrieve change rows one by one

```sql
-- Retrieve change rows one by one:
declare @from_lsn binary(10), @to_lsn binary(10), @max_lsn binary(10);
select @max_lsn =  sys.fn_cdc_map_time_to_lsn('largest less than or equal', GETDATE())
-- first row
select @from_lsn = sys.fn_cdc_get_min_lsn('dbo_t1')
select top 1 @from_lsn = __$start_lsn from CDC.dbo_t1_ct where __$start_lsn >= @from_lsn order by __$start_lsn
select @to_lsn = sys.fn_cdc_increment_lsn(@from_lsn)
select * from cdc.fn_cdc_get_all_changes_dbo_t1(@from_lsn, @to_lsn, 'all')
select @from_lsn as 'from_lsn', @to_lsn as 'to_lsn', @max_lsn as 'max_lsn'
-- next row
select top 1 @from_lsn = __$start_lsn from CDC.dbo_t1_ct where __$start_lsn > @from_lsn order by __$start_lsn
select @to_lsn = sys.fn_cdc_increment_lsn(@from_lsn)
select * from cdc.fn_cdc_get_all_changes_dbo_t1(@from_lsn, @to_lsn, 'all')
select * from cdc.fn_cdc_get_net_changes_dbo_t1(@from_lsn, @to_lsn, 'all')
select @from_lsn as 'from_lsn', @to_lsn as 'to_lsn', @max_lsn as 'max_lsn'
-- Check with:
select * from CDC.dbo_t1_ct
GO
```

This approach is retrieving the lower boundary as the first valid LSN from the capture instance (`sys.fn_cdc_get_min_lsn`) and immediately updates it with a valid LSN from the actual tracking table (`cdc.<capture_instance>_CT`). It avoids two scenarios: that the capture instance has a start_lsn that is lower than the first actual change in the tracking table; and that the tracking table has invalid, outdated LSNs because the cleanup is too slow and hasn't cleaned up the tracking table yet. The second valid LSN is read directly from the tracking table as the next after the first one.

The upper boundary is simply incremented based on the lower boundary so that it only retrieves a single change.

You might construct a While loop up to the @max_lsn following the same logic - try it out!

Retrieve all change rows up to a specific time

```sql
-- Use current time as cut-off time for retrieving change rows:
declare @from_lsn binary(10), @to_lsn binary(10), @save_to_lsn binary(10);
select @from_lsn = sys.fn_cdc_get_min_lsn('dbo_t1')
select top 1 @from_lsn = __$start_lsn from CDC.dbo_t1_ct where __$start_lsn >= @from_lsn order by __$start_lsn
select @to_lsn =  sys.fn_cdc_map_time_to_lsn('largest less than or equal', GETDATE())
select * from cdc.fn_cdc_get_all_changes_dbo_t1(@from_lsn, @to_lsn, 'all')
GO
```

This approach is retrieving the lower boundary as the first valid LSN from the capture instance (`sys.fn_cdc_get_min_lsn`) and immediately updates it with a valid LSN from the actual tracking table (`cdc.<capture_instance>_CT`). It avoids two scenarios: that the capture instance has a start_lsn that is lower than the first actual change in the tracking table; and that the tracking table has invalid, outdated LSNs because the cleanup is too slow and hasn't cleaned up the tracking table yet.

The upper boundary is the LSN that the capture job had logged close to the specified datetime value.

Schema change

```
-- perform schema change: add column
ALTER TABLE t1 ADD c2 NVARCHAR(100) NULL
GO
-- DDL shows up in the system tables:
exec sys.sp_cdc_get_ddl_history @capture_instance = 'dbo_t1'
select * from CDC.ddl_history
select * from dbo.systranschemas
-- but is not reflected in the current capture instance:
select * from CDC.captured_columns  -- what columns are tracked for each table
select * from CDC.dbo_t1_ct                     -- this is the actual tracking table
go
```

The stored procedure sys.sp_cdc_get_ddl_history is a wrapper for cdc.ddl_history, adding the clear-text details for the underlying table to its output.

The current capture instance remains unchanged and will not track any DML changes to the new column. If you want to track the new column, you need to create a new capture instance that includes the column in its column list. See the parameters for sys.sp_cdc_enable_table for more information.

Disable and remove CDC

```
-- Cleanup:

-- disable table
-- need to specify the capture instance, because one table might have several of them
use CDC_Publisher
exec sys.sp_cdc_disable_table @source_schema = 'dbo', @source_name = 't1', @capture_instance = 'dbo_t1'
go

-- disable database for CDC
use CDC_Publisher
exec sys.sp_cdc_disable_db
go

-- remove test database from instance
use master
go
drop database CDC_Publisher
go


/** End of Script **/
```

sys.sp_cdc_disable_table disables CDC for the specified source table and capture instance in the current database.

If you want to remove CDC completely, you can directly call sys.sp_cdc_disable_db without first executing `sys.sp_cdc_disable_table` - disabling the database removes all traces of CDC including system tables, tracking tables, and metadata.

Full script for convenient copy and paste

```sql
/** Start of Script **/

-- create database
CREATE DATABASE CDC_Publisher
GO
USE CDC_Publisher
-- enable database for CDC
if not exists (select * from master.sys.databases where name = 'CDC_Publisher' and is_cdc_enabled = 1)
        EXEC sys.sp_cdc_enable_db
go

-- create a table and add some data
USE CDC_Publisher
CREATE TABLE t1 (ID INT PRIMARY KEY NOT NULL, c1 VARCHAR(100))
GO
INSERT INTO t1 (ID, c1) VALUES (1, 'row one')
INSERT INTO t1 (ID, c1) VALUES (2, 'row two')
GO

-- enable table for CDC
EXEC sys.sp_cdc_enable_table  @source_schema = 'dbo', @source_name = 't1', @role_name = 'cdc_Admin'
GO
/*
Job 'cdc.CDC_Publisher_capture' started successfully.
Job 'cdc.CDC_Publisher_cleanup' started successfully.
*/

-- take a look at system tables and views:
select * from CDC.change_tables     -- details on active CDC tables
select * from CDC.captured_columns  -- what columns are tracked for each table
select * from CDC.index_columns     -- PK/unique column of tracked tables
select * from CDC.dbo_t1_ct         -- this is the actual tracking table
select * from CDC.lsn_time_mapping  -- mapping between LSNs and point in time
select * from CDC.ddl_history       -- DDL changes (only filled after capture job has run)
select * from dbo.systranschemas    -- DDL changes
go
-- display summary details through stored proc:
exec sp_cdc_help_change_data_capture @source_schema = 'dbo', @source_name = 't1'
go
-- display details on the capture and cleanup job:
exec sys.sp_cdc_help_jobs
select * from msdb..cdc_jobs
select * from msdb..sysjobs where name like 'cdc%'
go


-- Start capture job through code:
EXEC sp_MScdc_capture_job
-- This statement will remain in status "Executing query..."
-- Note that cancelling the execution in query window won't stop the actual job
-- Call sp_replflush to release the full logreader context:
exec sp_replflush
go

-- start and stop the SQL Agent capture job
exec sys.sp_cdc_start_job @job_type= 'capture'
exec sys.sp_cdc_stop_job @job_type= 'capture'

-- start and stop the SQL Agent cleanup job
exec sys.sp_cdc_start_job @job_type= 'cleanup'
exec sys.sp_cdc_stop_job @job_type= 'cleanup'


-- Apply some changes:
INSERT INTO t1 (ID, c1) VALUES (3, 'row three')
GO
UPDATE t1 SET c1 = 'update ' + c1 WHERE ID = 2
go
```

```
-- in the tracking table:
select * from CDC.dbo_t1_ct
-- will return rows only if the capture job was running!
-- will return 1 row for the insert:  __$operation = 2
-- will return 2 rows for the update: __$operation = 3 for the old data, = 4 for the new data

-- the wrapper functions will give you:
declare @from_lsn binary(10), @to_lsn binary(10)
set @from_lsn = sys.fn_cdc_get_min_lsn('dbo_t1')
set @to_lsn = sys.fn_cdc_get_max_lsn()
select * from cdc.fn_cdc_get_all_changes_dbo_t1(@from_lsn, @to_lsn, 'all')
select * from cdc.fn_cdc_get_net_changes_dbo_t1(@from_lsn, @to_lsn, 'all')


-- correlate LSN with a point in time (uses CDC.lsn_time_mapping internally)
declare @pit_lsn binary(10)
declare @lsn_pit datetime
-- get closest LSN for a datetime
select @pit_lsn = sys.fn_cdc_map_time_to_lsn('largest less than or equal', GETDATE())
select @pit_lsn, GETDATE()
-- get closest datetime for an LSN
select @lsn_pit = sys.fn_cdc_map_lsn_to_time(@pit_lsn)
select @pit_lsn as 'LSN from initial datetime', @lsn_pit as 'Datetime from LSN', GETDATE() as 'Initial datetim
go


-- Retrieve change rows one by one:
declare @from_lsn binary(10), @to_lsn binary(10), @max_lsn binary(10);
select @max_lsn =  sys.fn_cdc_map_time_to_lsn('largest less than or equal', GETDATE())
-- first row
select @from_lsn = sys.fn_cdc_get_min_lsn('dbo_t1')
select top 1 @from_lsn = __$start_lsn from CDC.dbo_t1_ct where __$start_lsn >= @from_lsn order by __$start_lsn
select @to_lsn = sys.fn_cdc_increment_lsn(@from_lsn)
select * from cdc.fn_cdc_get_all_changes_dbo_t1(@from_lsn, @to_lsn, 'all')
select @from_lsn as 'from_lsn', @to_lsn as 'to_lsn', @max_lsn as 'max_lsn'
-- next row
select top 1 @from_lsn = __$start_lsn from CDC.dbo_t1_ct where __$start_lsn > @from_lsn order by __$start_lsn
select @to_lsn = sys.fn_cdc_increment_lsn(@from_lsn)
select * from cdc.fn_cdc_get_all_changes_dbo_t1(@from_lsn, @to_lsn, 'all')
select * from cdc.fn_cdc_get_net_changes_dbo_t1(@from_lsn, @to_lsn, 'all')
select @from_lsn as 'from_lsn', @to_lsn as 'to_lsn', @max_lsn as 'max_lsn'
-- Check with:
select * from CDC.dbo_t1_ct
GO


-- Use current time as cut-off time for retrieving change rows:
declare @from_lsn binary(10), @to_lsn binary(10), @save_to_lsn binary(10);
select @from_lsn = MIN(__$start_lsn) from CDC.dbo_t1_ct
select @to_lsn =  sys.fn_cdc_map_time_to_lsn('largest less than or equal', GETDATE())
select * from cdc.fn_cdc_get_all_changes_dbo_t1(@from_lsn, @to_lsn, 'all')
GO


-- perform schema change: add column
ALTER TABLE t1 ADD c2 NVARCHAR(100) NULL
GO
-- shows up in the system tables:
select * from CDC.ddl_history
select * from dbo.systranschemas
-- but is not reflected in the current capture instance:
select * from CDC.captured_columns  -- what columns are tracked for each table
select * from CDC.dbo_t1_ct                     -- this is the actual tracking table
go


-- Cleanup:

-- disable table
-- need to specify the capture instance, because one table might have several of them
use CDC_Publisher
exec sys.sp_cdc_disable_table @source_schema = 'dbo', @source_name = 't1', @capture_instance = 'dbo_t1'
go

-- disable database for CDC
use CDC_Publisher
exec sys.sp_cdc_disable_db
go

-- remove test database from instance
use master
go
drop database CDC_Publisher
go


/** End of Script **/
```

How good have you found this content?