# Error 8510 Enlist operation failed

Last updated by | Radhika Shah | Dec 14, 2022 at 1:14 PM PST

Contents



## Issue

Customer receives error "Enlist operation failed" with error code 8510 when running distributed transactions.

This happens when a server instance fails to communicate with other instance while in the 'Enlistment Phase' (it happens while establishing a new connection to a database, .net SqlConnection.Open interface). First connection opened under TransactionScope is considered Transaction Manager and all subsequent connections under same scope to other databases will try to contact the TM (Transaction Manager) where this error can be seen.

It can happen for few reasons:

1. New connection belong to **different LogicalServer** which is different than Transaction Manager's logical server and user has **not established 'link'** between them to perform cross server transactions, please check if that is the case on this article: Investigating GT connectivity issues.
   If link is not an issue, then:
   1. We must check if error is permanent or transient. If error is permanent - than it can be a known issue where UCS communication layer is stuck in some funny state on target node causing communication break (one way communication). This can be resolved by killing the target instance. The issue is under investigation.
   2. If error is transient, we do not need any mitigation.
2. New connection belong to same LogicalServer, then we do not require any pre setup step. It may fail for same set of reasons mentioned above and we only need to mitigate for permanent error.

## Investigation/Analysis

### How to find target node

As described above, we need to find target instance which we like to kill as mitigation step, in below query, LogicalServerName of source or target may be same depending on if it is cross logical server transaction or not.

If it is cross logical server case, we need to place logical server name at right place below - we should not interchange source and target

```
MonGlobalTransactions
| where LogicalServerName == "qaa11-tenant-sqlserver"
| where message contains "InitForEnlist failed"
| where TIMESTAMP > ago(30m)
| extend SourceNodeName = NodeName
| extend SourceAppName = AppName
| extend transaction_id = unit_of_work_id
| summarize cnt() by transaction_id, SourceNodeName, SourceAppName
| join (
MonGlobalTransactions
| where LogicalServerName == "qaa11-tenant-sqlserver"
| where event == "dtc_transaction"
| where transaction_state == "Creating a new DTC transaction"
| where TIMESTAMP > ago(30m)
| extend TargetNodeName = NodeName
| extend TargetAppName = AppName
| extend transaction_id = unit_of_work_id
| summarize cnt() by transaction_id  , TargetNodeName , TargetAppName
) on transaction_id
```

For the above query, you may or may not want the TIMESTAMP filter which has been highlighted. You may keep/remove it at your own discretion.

The output will show target app name and node name. Using these details in XTS along with target Logical Server Name, we can pin point failing instance to be killed.

**Verify the error in telemetry, and if it's permanent or not**

User has reported the error so it better show up in our telemetry, this will be source node where error comes from. We may modify the query below to add TIMESTAMP if we know when error occurred. But below query will basically show AppName and NodeName of source node where error is showing up and it will also tell us how frequent it is by looking at cnt() output.

```
MonGlobalTransactions
| where LogicalServerName == "<Logical Server reported by user>"
| where message contains "InitForEnlist failed"
| summarize cnt() by NodeName, AppName
```

We can always change above query to zoom in to see when these errors occurred. We may filter by TIMESTAMP

```
MonGlobalTransactions
| where LogicalServerName == "<Logical Server reported by user>"
| where message contains "InitForEnlist failed"
```

Now how to know if error is permanent or not? Well, above query will give us last time when this error occurred, and we can use query below to know of any successful transaction happened after last error. In below query, we need to find Target Node or Transaction Manager's Logical Server Name, use above chapter **How to find target node**

If we see some rows here, it means user has successfully ran some transactions after the error occurred where Transaction Manager is same and same source participated in it. This will make this error transient. But if there

are no rows, we cannot assume error is permanent as user may not have ran any transactions after that error. So, we will have to ask user if he thinks error is permanent or not.

## Mitigation

- In scenarios where multiple instances participate, but those instances are part of the **same Subnet**, there are no additional requirements on a Subnet level (requirements from customer). However, instances are expected to use custom port for GT traffic from **range 11k-12k**.
- In scenarios where 2 or more instances from **different Subnets** participate in GT following requirements apply:

1. **Port 5024 needs to be open an all Subnets for all instances participating**. This port is used to initiate UCS connection for GT through Gateway.PDC (per-ring Gateway process for managing all TDS and UCS traffic).
2. **Port range 11k-12k need to be open on all Subnets inbound and outbound for all instances participating**. This range is used to allocate port for a single instance and use this port afterwards the UCS connection is established through Gateway.

- If the error is identified to be **permanent**, please raise ICM to *SQL Managed Instance: Distributed Transactions* team.

## Internal Reference

ICM 339657535 - Ports not open

## Public Doc Reference

Transactions for SQL Managed Instance

## Root Cause Classification

/Root Cause: Azure SQL v3/Surface Area/Distributed Transactions/DTC

**How good have you found this content?**