Distributed Transactions

Last updated by | Radhika Shah | Feb 27, 2023 at 6:43 PM PST

Contents

Self-help content presented in Azure Portal

(This content was shown to the customer during case submission. It's also visible on 'Diagnose and solve problems' blade.)

Distributed Transactions for Azure SQL Managed Instance let you run transactions that span several databases within multiple instances. When transactions cross managed instance boundaries, the participating instances need to be in a mutual security and communication relationship. You do this by creating a Server Trust Group.

Distributed Transactions are available for .NET applications using ADO.NET and integrate with the familiar programming experience using the System.Transaction classes. Additionally, for managed instance distributed transactions are available in Transact-SQL. For T-SQL or .NET distributed transactions across databases hosted only by managed instances, we recommend **native support for distributed transactions** as the approach.

Distributed Transaction Coordinator (DTC) for Azure SQL Managed Instance lets you run distributed transactions in **mixed environments** such as across managed instances, SQL Servers, other relational database management systems (RDBMSs), custom applications and other transaction participants hosted in any environment that can establish network connectivity to Azure.

Scan the following section headings and select one or more that will help resolve your issue.

Native support for distributed transactions across managed instances

This feature is relying on **Server Trust Groups** to establish and manage trust in cross-instance scenarios.

Distributed transactions that are initiated using either:

- BEGIN DISTRIBUTED TRANSACTION statement (from T-SQL code)
- Linked servers and linked server connectivity (from T-SQL code)
- Using TransactionScope class (from .NET code)

Troubleshooting steps for native support

In case of the error message, "MSDTC not available," use the following steps for connectivity and security:

- In scenarios with multiple instances, check the connectivity between the instances:
  - If distributed transaction with linked server is failing, try to execute a simple select query on the linked server. This query can indicate if the issue is with the linked server setup rather than distributed transactions. Basic requirement for linked servers: Ports 1433, 11000 - 12000 are allowed.
  - If instances are part of the same subnet, there are no additional requirements on the subnet level, and instances are expected to use custom port from range 11000-12000.
  - If instances are not part of the same subnet, port 5024 needs to be open on all subnets for all participating instances. Also, port range 11000-12000 needs to be open on all subnets for all participating instances.
  - If instances are on different VNets, check if VNet peering is properly set up.
- Check if security requirements are satisfied.
  - Verify that instances are added to the Server Trust Group.
- To resolve stuck, deadlocked, orphaned, or in-doubt distributed transactions, use the kill unit of work (or session id). For more details, see the KILL documentation.
- Make sure that you're running distributed transactions only across databases hosted on one or more Azure SQL Managed Instances and only with T-SQL or .NET (TransactionScope).

Distributed Transaction Coordinator (DTC) support for transactions in mixed environments

Troubleshooting steps for DTC in mixed environments

- Distributed Transaction Coordinator (DTC) for Azure SQL Managed Instance is currently in preview and is part of the November 2022 feature wave. To learn more about the timelines for feature wave rollout, see November 2022 feature wave.
- For T-SQL or .NET distributed transactions across databases hosted only by managed instances, we recommend native support for distributed transactions as the approach.
- To run distributed transactions, you need to:
  - Configure DTC
  - Enable network connectivity between participants
    - If the host/participant (in the external environment) is hosted on an Azure VNet, you can use VNet peering. Otherwise, establish connectivity by using your preferred method, such as point-to-site VPN, ExpressRoute, or another network connectivity technology that meets your business needs.
    - Ports 135 and 1024-65535 need to allow both inbound and outbound communication. This applies to both the Azure VNet network security group (NSG) for the managed instance, and any firewall on the external environment.
  - Configure DNS settings.
  - Test network connectivity between the DTC endpoints of your managed instance and the external DTC host.

Collab with On-prem team

When troubleshooting issues for MSDTC in mixed environment where the other environment is SQL On-premises, use this Support Area Path (SAP) to work in collaboration with On-prem team:

SQL Server/SQL Server 2022 on Windows/SQL Server 2022 Database Engine Windows/Startup, shutdown, restart issues (instance or database)/MSDTC Service issues

Public Blogs

- Cross-instance distributed transactions with Azure SQL Managed Instance (native support for distributed transactions)
- Distributed Transaction Coordinator for Azure SQL Managed Instance in mixed environments

Resources

- Distributed transactions across cloud databases - Native support
- Server Trust Group
- Distributed Transaction Coordinator (DTC) in mixed environments
- Connect applications to Azure SQL Managed Instance
- T-SQL differences between SQL Server & Azure SQL Managed Instance (distributed transactions)

How good have you found this content?