Data Integration Service Optimization and Stability
Abstract
You can configure the Data Integration Service to optimize stability or to optimize connection performance when it runs a task. To optimize stability, configure the Data Integration Service to run DTM instances in separate operating system processes. To optimize connection performance, configure connection pooling for database connections, and then configure the Data Integration Service to run DTM instances within one operating system process. This article explains how to configure the Data Integration Service for optimal stability or optimal connection performance.

Supported Versions
- Data Transformation 9.5.1 Hotfix 1
- Data Services 9.5.1 Hotfix 1
- Data Quality 9.5.1 Hotfix 1
- PowerExchange 9.5.1 Hotfix 1

Table of Contents
Data Integration Service Processes and Instances. ....................................................... 2
Connection Pooling. ............................................................................................. 3
  Connection Pooling Process. .............................................................................. 3
  Connection Pooling Example. ............................................................................ 3
  Pooling Properties in Connection Objects. .......................................................... 4
Data Integration Service Optimization. ................................................................. 4
  Optimize Stability ............................................................................................... 5
  Optimize Connection Performance. ................................................................. 5
PowerExchange Connection Pooling. ................................................................. 5
  PowerExchange Connection Pooling Process. .................................................... 6
  Connection Pooling for PowerExchange Netport Jobs. ...................................... 6
  PowerExchange Connection Pooling Configuration. ........................................... 7

Data Integration Service Processes and Instances
You can configure the Data Integration Service to run DTM instances in one operating system process or in separate operating system processes. A DTM instance is a specific, logical representation of the Data Transformation Manager. An operating system process is an instance of the operating system that is being executed.

Run DTM instances in one operating system process when most of the tasks that the Data Integration Service runs are short-running tasks. For example, profiles on small data sources are short-running tasks. All tasks run within one operating system process. You can optimize connection performance when you run DTM instances in one operating system process.

Run a DTM instance in separate processes when most of the tasks that the Data Integration Service runs can take a long time to complete. For example, a workflow is a long-running task. Running DTM instances in separate processes optimizes stability because any interruption to one task that the Data Integration Service runs has no effect on other tasks.

By default, the Data Integration Service runs DTM instances in separate processes.
Connection Pooling

Connection pooling is a framework to cache database connection information that is used by the Data Integration Service. Connection pooling increases performance through the reuse of cached connection information.

If connection pooling is disabled, the Data Integration Service creates a connection instance each time it processes an integration object. It drops the instance when it finishes processing the integration object.

If connection pooling is enabled, the Data Integration Service maintains a library of connection pools. A connection pool is a group of connection instances for one connection object. The connection pool library can contain multiple connection pools. The number of connection pools depends on the number of unique connections that the Data Integration Service uses while it runs tasks.

Each connection pool in the connection pool library contains connection instances for one connection object. A connection instance is a representation of a physical connection to a data source.

A connection instance can be active or idle. An active connection instance is a connection instance that the Data Integration Service is using to connect to a database. A Data Integration Service can create an unlimited number of active connection instances.

An idle connection instance is a connection instance in the connection pool that is not in use. The connection pool retains idle connection instances based on the pooling properties that you configure. You configure the minimum idle connections, the maximum idle connections, and the maximum idle connection time.

Connection Pooling Process

When the Data Integration Service runs a data integration task, it requests a connection instance from the pool. If an idle connection instance exists, the connection pool releases it to the Data Integration Service. If the connection pool does not have an idle connection instance, the Data Integration Service creates an active connection instance.

When the Data Integration Service completes the task, it releases the active connection instance to the pool as an idle connection instance. If the connection pool contains the maximum number of idle connection instances, the Data Integration Service drops the active connection instance instead of releasing it to the pool.

The Data Integration Service drops an idle connection instance from the pool when the following conditions are true:

- A connection instance reaches the maximum idle time.
- The connection pool exceeds the minimum number of idle connections.

When you start the Data Integration Service, the connection pool library drops all connections in the pool.

When you update the user name, password, or connection string for a database connection that has connection pooling enabled, the updates take effect immediately. Subsequent connection requests use the updated information. Also, the connection pool library drops all idle connections and restarts the connection pool. It does not return any connection instances that are active at the time of the restart to the connection pool when complete.

If you update any other database connection property, you must restart the Data Integration Service to apply the updates.

Connection Pooling Example

You want to use connection pooling to optimize connection performance.

You configure the following pooling parameters for a connection:

- Connection Pooling: Enabled
- Minimum Connections: 5
- Maximum Connections: 15
- Maximum Idle Time: 120 seconds
When the Data Integration Service receives a request to run 40 data integration tasks, it uses the following process to maintain the connection pool:

1. The Data Integration Service receives a request to process 40 integration objects at 11:00 a.m., and it creates 40 connection instances.
2. The Data Integration Service completes processing at 11:30 a.m., and it releases 15 connections to the connection pool as idle connections.
3. It drops 25 connections because they exceed the connection pool size.
4. At 11:32 a.m., the maximum idle time is met for the idle connections, and the Data Integration Service drops 10 idle connections.
5. The Data Integration Service maintains five idle connections because the minimum connection pool size is five.

**Pooling Properties in Connection Objects**

You can edit connection pooling properties in the **Pooling** view for a database connection.

The following list describes database connection pooling properties that you can edit in the **Pooling** view for a database connection:

**Enable Connection Pooling**

Enables connection pooling. When you enable connection pooling, the connection pool retains idle connection instance in memory. When you disable connection pooling, the Data Integration Service stops all pooling activity. To delete the pool of idle connections, you must restart the Data Integration Service.

**Minimum # of Connections**

The minimum number of idle connection instances that the pool maintains for a database connection. Set this value to be equal to or less than the idle connection pool size. Default is 0.

**Maximum # of Connections**

The maximum number of idle connections instances that the Data Integration Service maintains for a database connection. Set this value to be more than the minimum number of idle connection instances. Default is 15.

**Maximum Idle Time**

The number of seconds that a connection that exceeds the minimum number of connection instances can remain idle before the connection pool drops it. The connection pool ignores the idle time when it does not exceed the minimum number of idle connection instances. Default is 120.

**Data Integration Service Optimization**

You can configure the Data Integration Service to optimize stability or to optimize connection performance when it runs DTM instances.

To optimize stability, configure the Data Integration Service to run DTM instances in separate operating system processes. Running tasks in separate processes optimizes stability because any interruption to one task has no effect on any other task. By default, the Data Integration Service runs DTM instances in separate processes.

To optimize connection performance, configure the Data Integration Service to run DTM instances within one operating system process, and then configure connection pooling for the database connections. When multiple tasks run within a single process, the Data Integration Service can maintain and reuse a pool of database connections. Reusing connections optimizes performance because it minimizes the amount of time and resources that the Data Integration Service uses when it opens and closes multiple database connections.

If you run some tasks in which stability is most important and other tasks in which connection performance is most important, create multiple Data Integration Services. Configure one Data Integration Service to optimize stability and another to optimize connection performance.
Optimize Stability

To optimize stability, configure the execution options for the Data Integration Service to run tasks in separate operating system processes. Each task runs in a separate DTM process.

When the Data Integration Service runs DTM instances in separate processes, the DTM instances connect to the Data Integration Service process using the host name localhost. Therefore, the host file on the node that runs the Data Integration Service must contain a localhost entry. If the hosts file does not contain a localhost entry, mappings might fail.

Consider the following restrictions when you run DTM instances in separate processes:

- You cannot use connection pooling. If you configure connection pooling properties for the database connections, the Data Integration Service ignores them.
- You cannot limit the amount of memory that the Data Integration Service allocates to run requests. If you set the maximum memory size for the Data Integration Service process, the Data Integration Service ignores it.

To configure the Data Integration Service to optimize stability, enable Launch Jobs as Separate Processes in the Data Integration Service properties. When you configure the Data Integration Service to run DTM instances in separate processes, the Data Integration Service ignores the Maximum Memory Size property, DTM Keep Alive Time properties, and all pooling properties in the database connection properties.

Optimize Connection Performance

To optimize connection performance, configure the execution options for the Data Integration Service to run tasks within one operating system process. Then, configure connection pooling for the database connections or a keep alive time for the DTM instance.

When the DTM instances run in a single process, you can optimize connection performance through connection pooling. The Data Integration Service caches database connections for tasks and maintains a pool of connections that it can reuse. The Data Integration Service caches and releases the connections based on how you configure connection pooling properties for the connection.

The following list describes the options that you need to configure to optimize connection performance:

Connection Pooling

- Enable the Connection Pooling property in the database connection properties. Optionally, configure additional connection pooling properties.

Launch Jobs as Separate Processes

- Clear Launch Jobs as Separate Processes in the Data Integration Service properties.

Maximum Memory Size

- Optionally, configure Maximum Memory Size in the Data Integration Service process properties.

DTM Keep Alive Time

- Optionally, for SQL data services or web services, configure DTM Keep Alive Time in the SQL properties or web service properties for the Data Integration Service. Or, you can configure DTM Keep Alive Time for each deployed SQL data service or web service on the Applications view for the Data Integration Service. If you set this property for a deployed SQL data service or web service, the value for the deployed SQL data service or web service overrides the value you set for the Data Integration Service.

PowerExchange Connection Pooling

A PowerExchange connection pool is a set of network connections to a PowerExchange Listener. The Data Integration Service connects to a PowerExchange data source through the PowerExchange Listener.
PowerExchange uses connection pooling for the following types of database connection objects:

- Adabas
- DB2 for i5/OS
- DB2 for z/OS
- IMS
- Sequential
- VSAM

To define a connection to a PowerExchange Listener, include a NODE statement in the DBMOVER file on the Data Integration Service machine. Then define a database connection and associate the connection with the Listener. The Location property specifies the Listener node name. Define database connection pooling properties in the Pooling view for a database connection.

**PowerExchange Connection Pooling Process**

The Data Integration Service connects to a PowerExchange data source through the PowerExchange Listener. A PowerExchange connection pool is a set of connections to a PowerExchange Listener.

When the Data Integration Service runs a data integration task, it requests a connection instance from a connection pool. If the Data Integration Service requires a PowerExchange connection instance, it requests the connection instance from PowerExchange.

When PowerExchange receives a request for a connection to a Listener, it uses a connection in the pool that has matching characteristics, including user ID and password. If the pool does not contain a connection with matching characteristics, PowerExchange modifies and reuses a pooled connection to the Listener, if possible. For example, if PowerExchange receives a request for a connection for USER1 on NODE1 and finds only a pooled connection for USER2 on NODE1, PowerExchange reuses the connection, signs off USER2, and signs on USER1.

When PowerExchange returns a Listener connection to the pool, it closes any files or databases that the Listener had opened.

If you associate multiple database connection objects with the same Listener node name, PowerExchange combines the connections into a single pool. For example, if you associate multiple database connections to NODE1, a connection pool is used for all PowerExchange connections to NODE1. To determine the maximum size of the connection pool for the Listener, PowerExchange adds the Maximum # of Connections values that you specify for each database connection that uses the Listener.

If you want each database connection object to use a separate connection pool, define multiple NODE statements for the same PowerExchange Listener and associate each database connection object with a different Listener node name.

**Note:** PowerExchange connection pooling does not reuse netport connections unless the user name and password match.

**Connection Pooling for PowerExchange Netport Jobs**

Netport jobs that use connection pooling might result in constraint issues.

Depending on the data source, the netport JCL might reference a data set or other resource exclusively. Because a pooled netport connection can persist for some time after the data processing has finished, you might encounter concurrency issues. If you cannot change the netport JCL to reference resources nonexclusively, consider disabling connection pooling.

In particular, IMS netport jobs that use connection pooling might result in constraint issues. Because the program specification block (PSB) is scheduled for a longer period of time when netport connections are pooled, resource constraints can occur in the following cases:

- A netport job on another port might try to read a separate database in the same PSB, but the scheduling limit is reached.
The netport runs as a DL/1 job, and you attempt to restart the database within the IMS/DC environment after the mapping finishes running. The database restart fails, because the database is still allocated to the netport DL/1 region.

Processing in a second mapping or a z/OS job flow relies on the database being available when the first mapping has finished running. If pooling is enabled, there is no guarantee that the database is available.

You might need to build a PSB that includes multiple IMS databases that the Data Integration Service accesses. In this case, resource constraint issues are more severe as netport jobs are pooled that tie up multiple IMS databases for long periods. This requirement might apply because you can include up to ten NETPORT statements in a DBMOVER file. Also, PowerExchange data maps cannot include program communication block (PCB) and PSB values that PowerExchange can use dynamically.

**PowerExchange Connection Pooling Configuration**

To configure PowerExchange connection pooling, include statements in the DBMOVER configuration files on each machine that hosts the PowerExchange Listener or the Data Integration Service. Also, define connection pooling properties in the **Pooling** view of the connection.

**DBMOVER Configuration Statements for PowerExchange Connection Pooling**

To configure PowerExchange connection pooling, define DBMOVER configuration statements on each machine that hosts the PowerExchange Listener or the Data Integration Service.

Define the following statements:

**LISTENER**

Defines the TCP/IP port on which a named PowerExchange Listener process listens for work requests. Include the LISTENER statement in the DBMOVER configuration file on the PowerExchange Listener machine.

**MAXTASKS**

Defines the maximum number of tasks that can run concurrently in a PowerExchange Listener. Include the MAXTASKS statement in the DBMOVER configuration file on the PowerExchange Listener machine.

Ensure that MAXTASKS is large enough to accommodate twice the maximum size of the connection pool for the Listener. The maximum size of the connection pool is equal to the sum of the values that you enter for the Maximum # of Connections pooling property for each database connection that is associated with the Listener.

Default is 30.

**NODE**

Defines the TCP/IP host name and port that PowerExchange uses to contact a PowerExchange Listener. Include the NODE statement in the DBMOVER file on the Data Integration Service machine.

**TCPIP_SHOW_POOLING**

Writes diagnostic information to the PowerExchange log file. Include the TCPIP_SHOW_POOLING statement in the DBMOVER file on the Data Integration Service machine.

If TCPIP_SHOW_POOLING=Y, PowerExchange writes message PWX-33805 to the PowerExchange log file each time a connection is returned to a PowerExchange connection pool.

Message PWX-33805 provides the following information:

- Size. Total size of PowerExchange connection pools.
• Hits. Number of times that PowerExchange found a connection in a PowerExchange connection pool that it could reuse.
• Partial hits. Number of times that PowerExchange found a connection in a PowerExchange connection pool that it could modify and reuse.
• Misses. Number of times that PowerExchange could not find a connection in a PowerExchange connection pool that it could reuse.
• Expired. Number of connections that were discarded from a PowerExchange connection pool because the maximum idle time was exceeded.
• Discarded pool full. Number of connections that were discarded from a PowerExchange connection pool because the pool was full.
• Discarded error. Number of connections that were discarded from a PowerExchange connection pool due to an error condition.

Pooling Properties in PowerExchange Connection Objects

Configure connection pooling properties in the Pooling view for a PowerExchange database connection.

Enable Connection Pooling

Enables connection pooling. When you enable connection pooling, the connection pool retains idle PowerExchange Listener connection instances in memory. When you disable connection pooling, the Data Integration Service stops all pooling activity. To delete the pool of idle connections, you must restart the Data Integration Service.

Default is enabled for DB2 for i5/OS and DB2 for z/OS connections. Default is disabled for Adabas, IMS, Sequential, and VSAM connections.

Minimum # of Connections

The minimum number of idle connection instances that the pool maintains for a database connection. If multiple database connections are associated with a PowerExchange Listener, PowerExchange determines the minimum number of connections to the PowerExchange Listener by adding the values for each database connection.

Maximum # of Connections

The maximum number of idle connections instances that the pool maintains for a database connection. If multiple database connections are associated with a PowerExchange Listener, PowerExchange determines the maximum number of connections to the PowerExchange Listener node by adding the values for each database connection.

Verify that the value of MAXTASKS in the DBMOVER configuration file is large enough to accommodate twice the maximum number of connections to the PowerExchange Listener node.

Enter 0 to specify an unlimited connection pool size.

Default is 15.

Maximum Idle Time

The number of seconds that a connection that exceeds the minimum number of connection instances can remain idle before the connection pool drops it. The connection pool ignores the idle time when it does not exceed the minimum number of idle connection instances.

If multiple database connections are associated with a PowerExchange Listener, PowerExchange calculates the arithmetic mean of the non-zero values for each database connection to determine the maximum idle time for connections to the same Listener.

Default is 120.

Tip: Assign the same maximum idle time to each database connection.
Authors

Lori Troy
Senior Technical Writer

Ashlee Brinan
Senior Documentation Manager

Acknowledgements

The authors would like to acknowledge Pritesh Maker, Kiran Mehta, and Gautam Mudurni for their assistance with this article.