Getting Started with MS SQL Server and ThingWorx

Version 1.1

PTC Inc., 140 Kendrick Street, Needham, MA 02494 USA

Contents

Document Revision History 4
Overview 5
SQL Server Editions Comparison 5
Which Edition Should I use for a Production Server? 6
Prerequisites 6
MS SQL Server Requirements 6
Hardware and Software Requirements 6
Operating System Requirements 6
ThingWorx Connectivity to MS SQL Server 6
Planning for MS SQL Installation 7
Environment 7
On-Premise 7
Amazon EC2 7
Microsoft Windows Azure SQL Database 9
Capacity Planning 9
MS SQL Database Installation and Configuration 10
Installing MS SQL Server 2014 on Database Server 11
Installing SQL Server Management Studio (Database Admin Client) on Windows Machine 14
Connecting to the Client with the Database Server (Windows) 16
Installing the ODBC Driver for SQL Server (Linux) 16
Connecting to the Database Server (Linux) 17
Database Setup for ThingWorx Platform 18
With respect to ThingWorx, the twadmin login will be created manually by SSMS. The remaining steps are performed by executing bat (Windows) and shell (Linux) scripts via the command line. 18
Database Setup (Windows) 18
Creating the twadmin login 18
Configuring and Executing the Database Setup Script (Windows) 18
Configuring and Executing the Model/Data Provider Schema Script (Windows) 19
Configuring and Executing the Database Cleanup Script (Windows) 20
Database Setup (Linux) 21
Creating login **twadmin** 21

Configuring and Executing the Database Setup Script (Linux) 21
Configuring and Executing the Model/Data Provider Schema Script (Linux) 22
Configuring and Executing the Database Cleanup Script (Linux) 23
Configuring ThingWorx for MS SQL Server 24
Installing the MS SQL JDBC Driver 24
MS SQL Server Configuration Options in ThingWorx 24
Performance Monitoring and Tuning Tools in MS SQL Server 30
Microsoft SQL Server Native Backup and Restore Support 30
Scaling Out SQL Server 31
Scalable Shared Databases 31
Index Size Limitation and Implementation 31
Appendix: Sample platform-settings. json 32

Document Revision History

Revision Date	Version	Description of Change
June 08, 2017	1.1	Added a section for index size limitation.
March 22, 2017	1.0	Initial document version.

Getting Started with MS SQL Server and ThingWorx

Overview

ThingWorx provides the option to choose a persistence provider for your value stream, stream, and data table data. In ThingWorx 7.4, MS SQL Server is another persistence provider option in ThingWorx.

SQL Server is a relational database management system developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications—which may run either on the same computer or on another computer across a network (including the Internet).

SQL Server Editions Comparison

There are several SQL Server editions that you can choose from to best fit your data solution. The table below compares the three choices for editions for the MS SQL Server.

Feature	Enterprise Edition	Standard Edition	Express Edition
Maximum relational Database size	524 PB	524 PB	10 GB
Maximum memory utilized (per instance of SQL Server Database Engine)	Operating system maximum	128 GB	1 GB
AlwaysOn Availability Groups	Yes	No	No
Backup compression	Yes	Yes	No
Database Mirroring	Yes	Yes	Witness only
Log Shipping	Yes	Yes	No
Multi-instance support	50	50	50
Encrypted Backup	Yes	Yes	No
Table and index partitioning	Yes	No	No
Parallel query processing on partitioned tables and indices	Yes	No	No

For detailed descriptions about features supported by different editions of SQL Server 2014, refer to the following:

- https://msdn.microsoft.com/en-us/library/cc645993(v=sql.120).aspx
- https://msdn.microsoft.com/en-us/library/ms144275(v=sql.120).aspx

Which Edition Should I use for a Production Server?

Generally, the SQL Standard Edition is suitable for production environments, as it supports most of the features required.
If your production environment requires High Availability features such as AlwaysOn and other features such as In-Memory OLTP, Table and index partitioning, SQL Enterprise Edition is recommended.

Prerequisites

ThingWorx 7.4 or later

MS SQL Server Requirements

Hardware and Software Requirements

MS SQL Server 2014 or later

The minimum requirements for MS SQL Server 2014 Express Edition are located at:
https://msdn.microsoft.com/en-in/library/ms143506(v=sql.120).aspx

For SQL Server Sizing Resources for production, refer to the Capacity Planning section of this document.

Operating System Requirements

Supported operating systems for MS SQL Server is Microsoft Windows only. Linux may be supported in the future. The choice of the operating system does not affect ThingWorx. For more information, refer to the following:

https://msdn.microsoft.com/en-in/library/ms143506(v=sql.120).aspx

ThingWorx Connectivity to MS SQL Server

ThingWorx connects to the MS SQL database using a SQL JDBC driver. For more information on JDBC connection properties, refer to the following:

https://technet.microsoft.com/en-us/library/ms378988(v=sql.105).aspx

These connection properties can be configured the **platform-settings.json** file that is available in the ThingWorx software download. The **platform-settings.json** file contains the following connection properties:

jdbcUrl: jdbc:sqlserver://localhost:1433;databaseName=thingworx;applicationName=Thingworx;"

Connection properties:

ServerName - The computer running SQL Server. – *localhost*
PortNumber - The port where SQL Server is listening. -*1433*
DatabaseName -The name of the database to connect to **–** *thingworx*
applicationName- The application name - *Thingworx*

Planning for MS SQL Installation

Environment

The MS SQL Installation can be installed in the following environments:

- On-Premise
- Amazon EC2
    - SQL Server in Amazon EC2
    - Microsoft SQL Server on Amazon RDS
- Microsoft Azure

On-Premise

The SQL Server Installation Wizard provides a single feature tree for installation of all SQL Server components so that you do not have to install them individually.

For more information, refer to https://msdn.microsoft.com/en-us/library/ms143219(v=sql.120).aspx

Amazon EC2

SQL Server implementation in Cloud:

- SQL Server in Amazon EC2
- Microsoft SQL Server on Amazon RDS

SQL Server in Amazon EC2

- Amazon Web Services offers you the flexibility to run Microsoft SQL Server for as much or as little time as you need. You can select from a number of versions and editions, as well as choose between running it on Amazon Elastic Compute Cloud (Amazon EC2) or Amazon Relational Database Service (Amazon RDS).
- Using SQL Server on Amazon EC2 gives you complete control over every setting, similar to when it's installed on-premises.

Amazon EC2 Environment Specifics

Use the following facilities in Amazon EC2 for production environments:

- Dedicated hardware

- Cross Availability Zones network latency can be six times higher than intra-zone network latency. Therefore, all the machines in the "Zone A – Data Center" should reside in a single Availability Zone.
- Different Availability Zones and/or a Region for the "Zone B- Data Center" for disaster recovery. Each Region is a separate geographic area and has multiple, isolated locations known as Availability Zones. For more information, reference http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html
- Amazon EC2 VPC. Deploying machines into a Virtual Private Network to get complete control over the virtual networking environment, including a selection of your own IP addresses range, creation of subnets, and configuration of route tables and network gateways. For more information, reference http://aws.amazon.com/vpc/
    - A placement group is a logical grouping of Amazon EC2 instances within a single Availability Zone. Using placement groups enables applications to participate in a low-latency, 10 Gbps network. For more information, reference http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/placement-groups.html
    - Instances launched into a common cluster placement group are placed into a logical cluster that provides high-bandwidth, low-latency networking between all instances in the cluster. C4, C3, I2, CR1, G2, and HS1 instances support cluster networking. M3 instances do not. http://aws.amazon.com/ec2/instance-types/
    - Multiple Elastic Network Interfaces (ENI) with multiple IP addresses in the same VPC as the ThingWorx Platform server http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-eni.html
- Hardware Virtual Machine (HVM) uses a new network virtualization stack that provides higher I/O performance and lower CPU utilization compared to traditional implementations. In order to take advantage of Enhanced Networking, an HVM AMI should be launched in a VPC, and install the appropriate driver.
- SSD Ephemeral Storage (EBS volumes are not recommended). SSD-backed instance storage is optimized for very high random I/O performance.
- S3 Simple Storage Service for backups https://console.aws.amazon.com/s3/home?region=us-east-1#

A guide to deploy MS SQL Server on AWS: https://aws.amazon.com/windows/products/sql/

Configuring a SQL Server AlwaysOn Availability Group

Refer to the following for more information:
http://docs.aws.amazon.com/quickstart/latest/sql/part3.html

Microsoft SQL Server on Amazon RDS

Amazon RDS for SQL Server makes it easy to set up, operate, and scale SQL Server deployments in the cloud. With Amazon RDS, you can deploy multiple editions of SQL Server (2008 R2, 2012, 2014 and

2016) including Express, Web, Standard and Enterprise, in minutes with cost-efficient and re-sizable compute capacity.

For more information, refer to the following link:

http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_SQLServer.html

For a quick guide to deploy MS SQL, refer to the following for more information:

https://d0.awsstatic.com/whitepapers/RDS/Deploying_SQLServer_on_AWS.pdf

SQL Server with Microsoft Windows Server Failover Clustering (WSFC) clusters on AWS cloud, refer to:

http://docs.aws.amazon.com/quickstart/latest/sql/welcome.html

For the high availability option, refer to the following links:

https://aws.amazon.com/about-aws/whats-new/2014/05/19/amazon-rds-for-sqlserver-introduces-multi-az-support/

http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.MultiAZ.html

DB Instance Class Support for Microsoft SQL Server:

The computation and memory capacity of a DB instance is determined by its DB instance class. The DB instance class you need depends on your processing power and memory requirements. For more information, see DB Instance Class.

http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_SQLServer.html#SQLServer.Concepts.General.InstanceClasses

Microsoft Windows Azure SQL Database

Microsoft Azure SQL Database is a managed cloud database for app developers that makes building and maintaining applications easier and more productive.

SQL Azure enables organizations to store relational data in the cloud and quickly scale the size of their databases up or down as business needs change. Data is hosted, managed, and provisioned in Microsoft data centers.

Organizations can build applications on-premises and move them to SQL Azure or build them on Windows Azure and keep the data in the cloud. SQL Azure supports SQL Server's Transact-SQL (T-SQL) query language, offers built-in support for high availability and fault tolerance and allows for data to be synchronized between on-premises SQL Server and cloud databases.

Refer to the following link for more details:

https://docs.microsoft.com/en-us/azure/sql-database/

Capacity Planning

To determine if MS SQL Server is the right solution for your data, refer to the sizing and planning sections from the following Microsoft documentation:

https://blogs.msdn.microsoft.com/bartd/2010/06/16/sql-server-sizing-resources/

https://msdn.microsoft.com/en-us/library/jj874401.aspx

https://msdn.microsoft.com/en-us/library/ms143506(v=sql.120).aspx

Understand SQL Server and IOPS
When configuring a new server for SQL Server or when adding or modifying the disk configuration of an existing system, it is good practice to determine the capacity of the I/O subsystem prior to deploying SQL Server.

Before installing SQL, we recommend that you benchmark the I/O subsystem by using the SQLIO disk subsystem benchmark tool.

For information about how to use the SQLIOSim utility and SQLIO for stress testing, refer to the TechNet video Stress testing using SQLIOSIM and SQLIO.

Choose disk types

The disk types that you use in the system can affect reliability and performance.

Use solid state drives (SSD) for storage in SQL Server.

Choose RAID types

Although RAID is not a part of SQL Server, implementing RAID can directly affect the way SQL Server performs. RAID levels 0, 1, and 5 are typically used with SQL Server.

Compute Capacity Limits by Edition of SQL Server

To learn more about Compute Capacity Limits, refer to the following:

https://technet.microsoft.com/en-us/library/ms143760(v=sql.120).aspx

Maximum Capacity Specifications for SQL Server

To learn more about Maximum Capacity Limits, refer to the following:

https://technet.microsoft.com/en-us/library/ms143432(v=sql.120).aspx

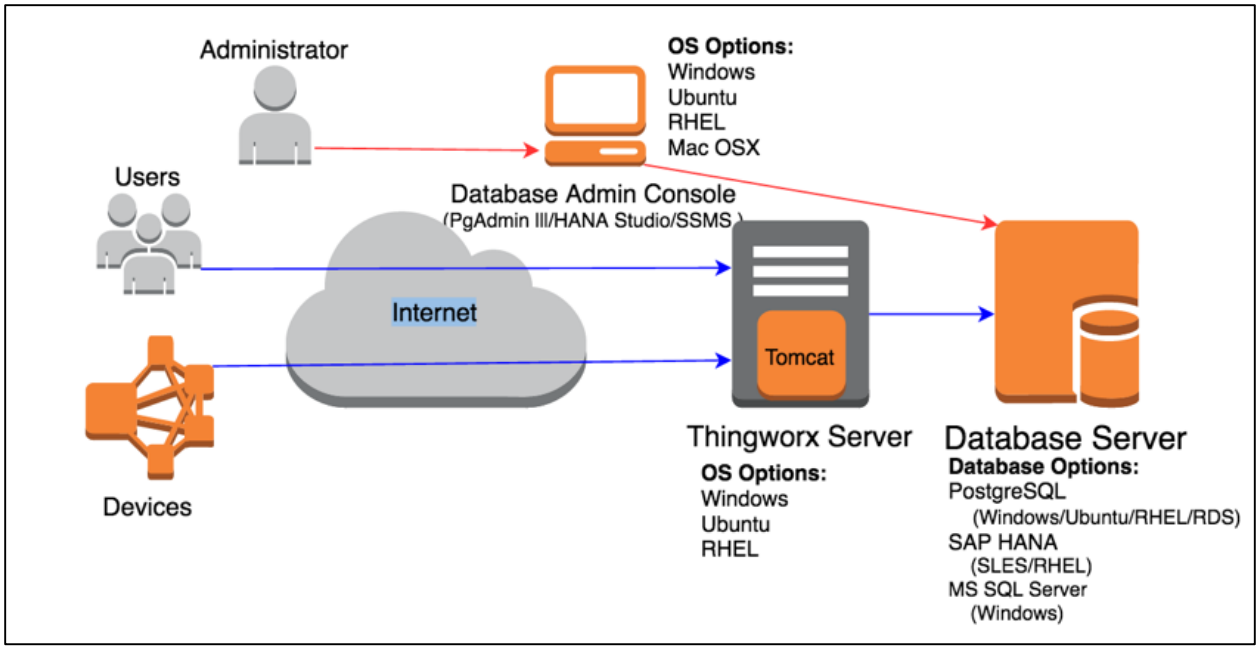
Pre-deployment I/O Best Practices

For SQL Server best practices, refer to the following article:

https://technet.microsoft.com/library/Cc966412

MS SQL Database Installation and Configuration

The architecture diagram below shows the options for ThingWorx Platform and database setup:



Installing MS SQL Server 2014 on Database Server

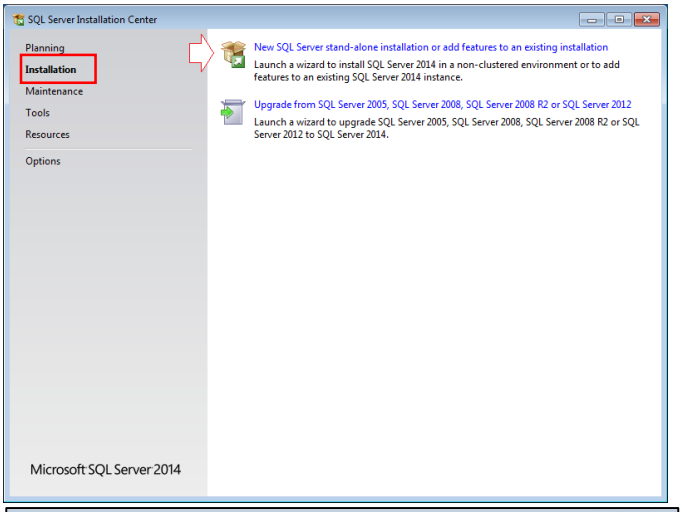
1. Obtain the licensed MS SQL Server 2014 installation from your software vendor for production deployments. The free express edition can be found at: http://www.microsoft.com/en-us/download/details.aspx?id=42299 for development and evaluation purposes.
2. After obtaining the installation software, double click on the installable file. The SQL Server installation center window will open.

   NOTE: For highly available production installations, follow this guide: https://msdn.microsoft.com/en-us/library/hh231721(v=sql.120).aspx

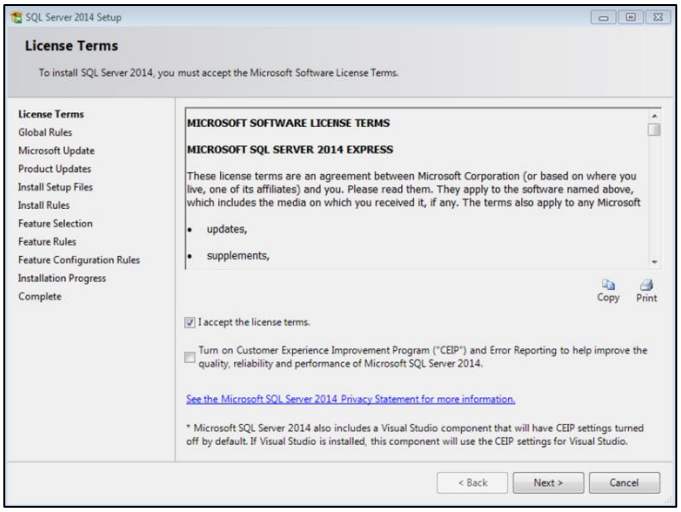
   The following list of customizations are recommended while installing the MS SQL Server to be used with ThingWorx:

   - **Named instance** – You can give any name for the instance or can keep it default. Make a note of it because this instance name will be used later to execute scripts on the database server.
   - **Mixed mode authentication** – Mixed mode enables both Windows Authentication and SQL Server Authentication. Windows authentication is using the account with which you have logged into the machine. Windows authentication is more secure. If it does not work during DBSetup script execution due to any domain issue, you can use the Server Authentication account. Always use a very strong password for the Server Authentication account.
   - **Data directories** – The best practice for choosing data directories is to keep data file directories, tempdb directories, log directories, and backup directories on separate physical hard disk drives, raided disk array, or a SAN.
   - **Connectivity** – Enable TCP/IP Protocol in Sql Server Configuration Manager

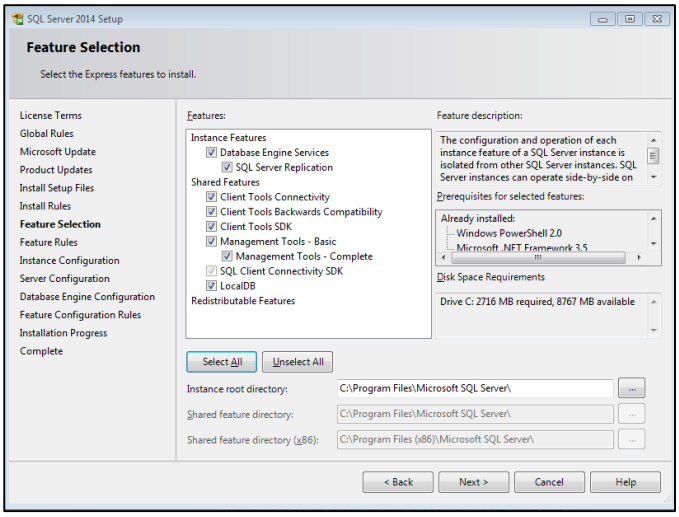
3. Select the Installation tab in the top-left corner of the SQL Server Installation Center.
4. Click **New SQL Server stand-alone installation or add features to an existing installation**.



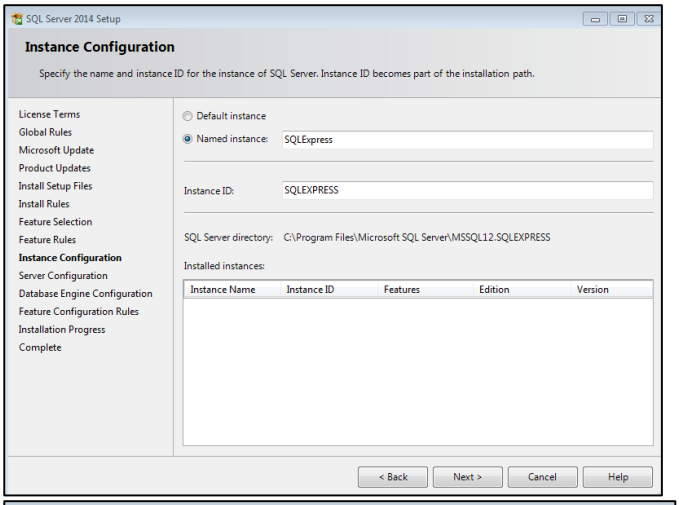
5. Accept the license terms and click **Next**.



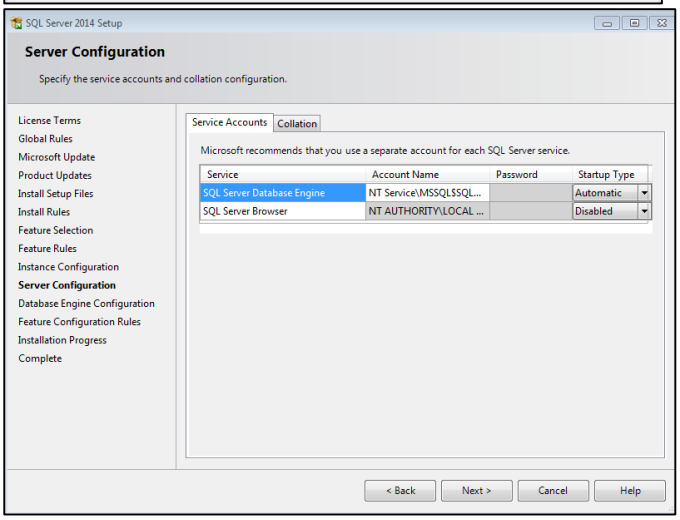
6. Select the features according to your requirements or **Select All**.
7. If necessary, in the **Instance root directory** field, you can specify the path of the installation directory.
8. Click **Next**.



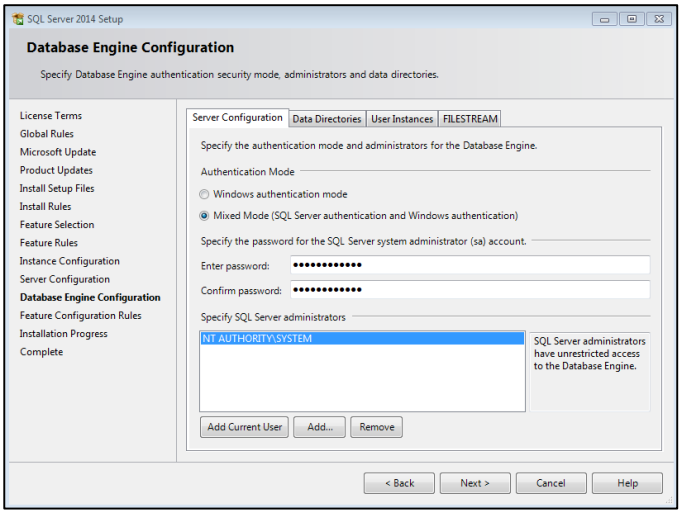
9. Specify the name of the SQL instance or keep it default.
   NOTE: This SQL Instance name will be used later for running the batch scripts for database setup. Keep note of its name for later use.
10. Click **Next**.



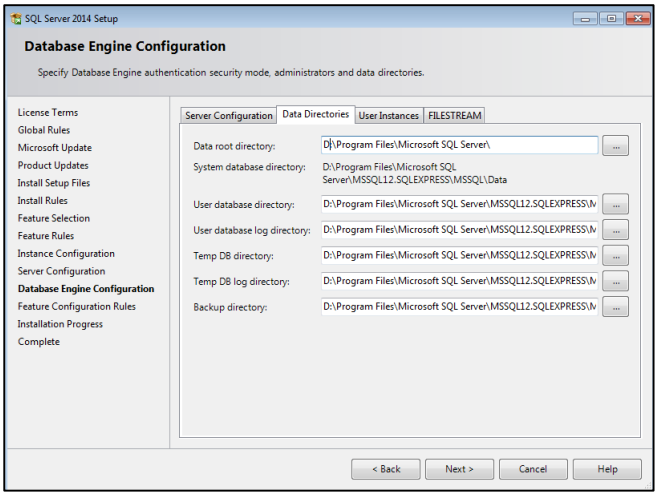
11. On the **Server Configuration** page, you can select the service accounts for running specific services of SQL. For ThingWorx, keep them as default.
12. Click **Next**.



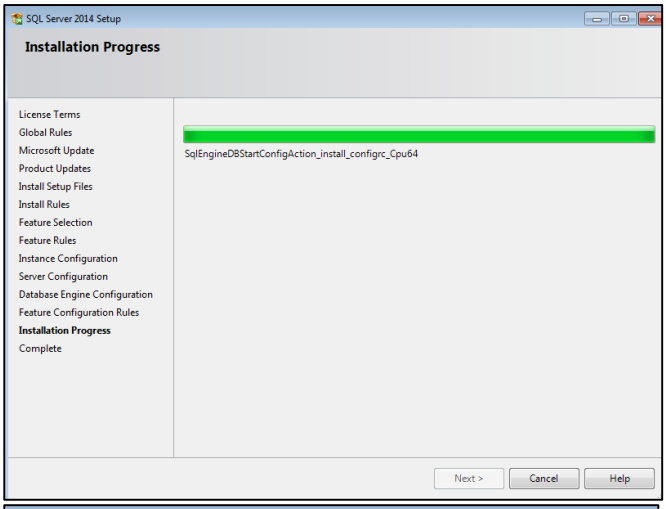
13. On the **Database Engine Configuration** page, click the **Server Configuration** tab.
14. Select **Mixed Mode** and provide a password for the “sa” account. If necessary, you can also add any other accounts (such as current Windows login account) in the **SQL Server administrators** group.



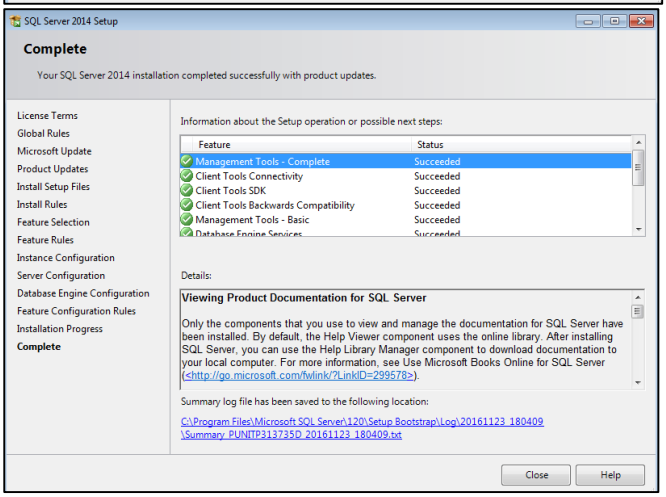
15. In the **Data Directories** tab, you can specify the directories and path for storing the data, log, temp, backup files according to your requirements and storage space availability.
16. Click **Next**.



17. Installation starts. After completion, click **Next**.



18. You can check the installation status of each feature in the **Complete** window. Click **Close**.

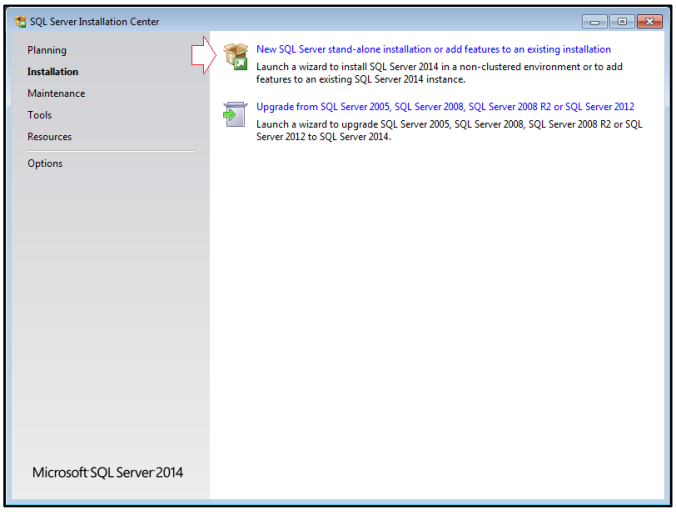


Installing SQL Server Management Studio (Database Admin Client) on Windows Machine

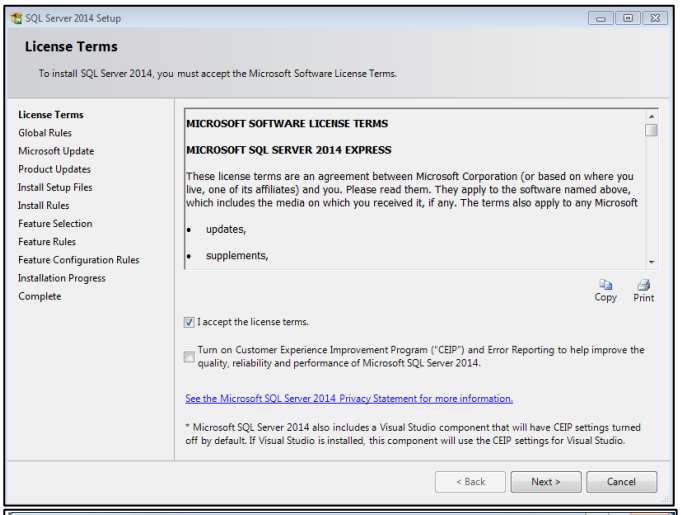
NOTE: SQL Server Management Studio (SSMS) is an integrated environment for accessing, configuring, managing, administering, and developing all components of SQL Server. Installing SSMS is only required if the administrator does not have access to the database server machine or if you want to separate the

client to connect to the database server. Skip this section if a separate client is not necessary.

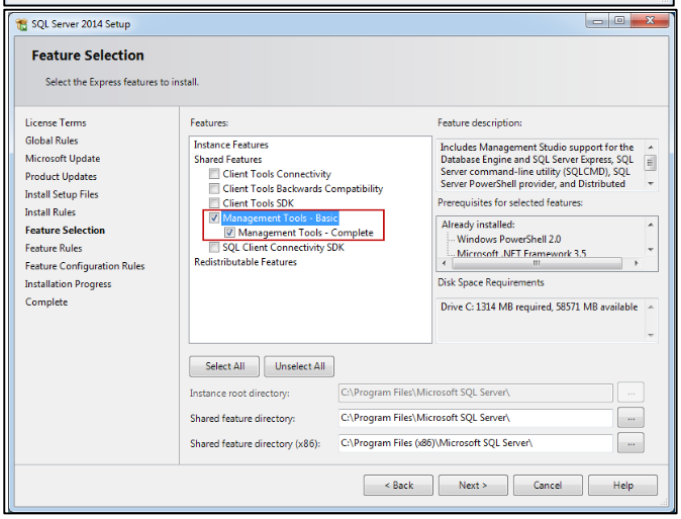
1. Select the **Installation** tab in the top-left corner of the SQL Server Installation Center.
2. Click **New SQL Server stand-alone installation or add features to an existing installation**.



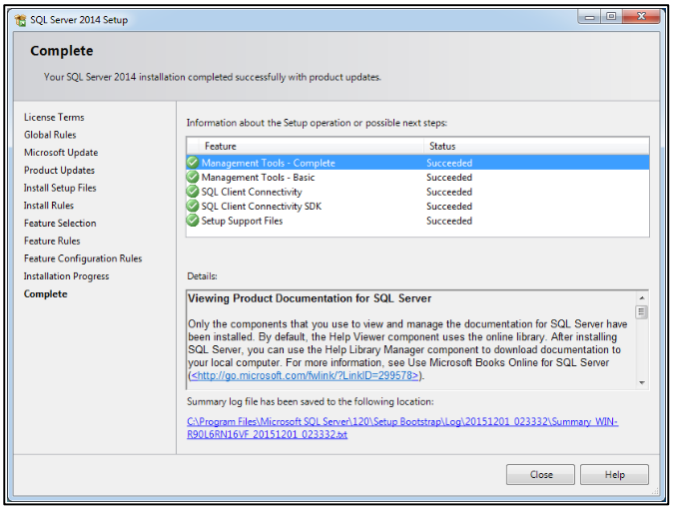
3. Accept the license terms and click **Next**.



4. Select **Management Tools – Basics** and the sub item **Management Tools - Complete**.
   NOTE: The SQL Client Connectivity SDK will be installed by default, whether you checked it or not.
5. Click **Next**.



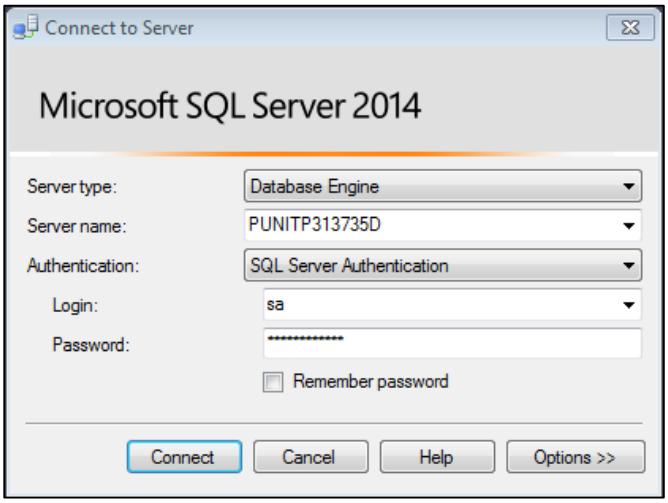
6. The summary for all the installed features and status of installation for each one, is shown. Click **Close**.



Connecting to the Client with the Database Server (Windows)

Skip this section if you have not installed SSMS.

1. Open SSMS from the start menu.
2. Enter the following details:
   - **Server type:** Database Engine
   - **Server name**: FQDN or IP of the database server.
   - **Authentication**: Any authentication mode (Windows authentication/SQL Server Authentication)
   - **Login**: sa (in case of SQL Server Authentication
   - **Password**: Password of sa account



3. Click **Connect**.
   NOTE: In the case of Windows authentication, SQL automatically takes the Windows credentials.

Installing the ODBC Driver for SQL Server (Linux)

NOTE: This section is only required if the administrator does not have access to the database server machine or if there is a requirement to separate the client to connect to the database server. Skip this section if the ThingWorx database setup scripts need to be executed from a Linux machine. If the

"Database Admin Console" is a Windows machine, as per the diagram above, then this section can be skipped.

For more information, refer to the following for installation of ODBC Driver for SQL Server on Linux:

https://docs.microsoft.com/en-us/sql/connect/odbc/linux/installing-the-microsoft-odbc-driver-for-sql-server-on-linux

Connecting to the Database Server (Linux)

Accessing a SQL Server ODBC Driver Data Source

If you did not create a data source during the installation, you must create one. The SQL Server ODBC driver installation creates a sample data source named **SQLSERVER_SAMPLE** that you can use as a starting point.

1. As root, **open /etc/odbc.ini** in a text editor.
2. To locate the sample data source, search for **[SQLSERVER_SAMPLE].**
3. Change the following attribute values:
    - **Server**: The host name (or IP address) of the machine where your SQL Server instance is running. To connect to a named instance, use the format **machinename\instancename**. To connect to a SQL Server Express instance, use the format **machinename\SQLEXPRESS**.
    - **Port**: If the SQL Server instance is listening on the default port, leave this set to **1433**. If your database administrator told you to specify a different port, replace 1433 with the new port number. Otherwise, delete 1433.
    - **Database**: The name of database to connect.
    - **User**: Your SQL Server login name. If you usually connect to SQL Server through your Windows account, use your Windows user name. If the SQL Server instance is running on a machine that is part of a Windows domain, use the format domain\username. Otherwise, type a valid SQL Server user name.
    - **Password**: The password for the login name specified by User.
4. Use isql to test the new data source:

    ```
    cd /usr/local/easysoft/unixODBC/bin
    ./isql -v dsn_name
    ```

    Where `dsn_name` is name of your ODBC data source. If you created a data source during the SQL Server ODBC driver installation, specify that data source name. If you have just edited the sample SQL Server ODBC driver data source, type SQLSERVER_SAMPLE.
5. At the prompt, type a select statement or type **help** to display a list of tables. To exit, press **RETURN** in an empty prompt line.

Database Setup for ThingWorx Platform

In MS SQL Server, a login needs to be created before creating a user. The login is created outside the database. Then, a database will be created. Inside the database, a user will be created and assigned the login created. Then, a schema needs to be created inside the database. Finally, the owner of the schema will be the user created above.

With respect to ThingWorx, the twadmin login will be created manually by SSMS. The remaining steps are performed by executing bat (Windows) and shell (Linux) scripts via the command line.

Database Setup (Windows)

NOTE: If you are not using Windows, go to the Linux section.

NOTE: Before executing the following steps, verify that the sections Installing SSMS on Windows Client and Connecting with the Database Server (Windows) are performed.

Creating the twadmin login

Using GUI

1. In SQL Server Management Studio (SSMS), open Object Explorer and expand the folder of the server instance in which to create the new login.
2. Right-click the **Security** folder, point to **New**, and click **Login**.
3. On the **General** page, enter a name for the new login in the **Login name** box.
4. Select **SQL Server Authentication**.
5. Enter a password for the login.
6. Click **OK**.

Using T-SQL command

In Query Editor in SSMS, enter the following Transact-SQL command:

```
CREATE LOGIN twadmin WITH PASSWORD = '<password> ';
```

Configuring and Executing the Database Setup Script (Windows)

1. Obtain the database setup script (**thingworxMssqlDBSetup.bat**) from the ThingWorx software download.
2. Open command prompt and change the directory to script folder.
3. Enter the below command:

```
thingworxMssqlDBSetup.bat -h <server> -i <server-instance> -p
<port> -a <database-admin-user-name> -l <login-name> -d
<thingworx-database-name> -u <thingworx-user-name> -s <schema-
name>
```

4. Upon execution, it will ask for password of database-admin-user. Enter the password and click enter. The following tasks will be performed by this bat script:

Once the script is executed, a database will be created with a user inside the database and a login associated with the user. This user will be assigned the **db_owner** role on created database. A schema will also be created if provided on the command line. Authorization of this schema will be given to the created user.

Description of Parameters:

Option	Parameter	Default	Description	Example
-h	server	localhost	FQDN or IP of database server	-h 10.0.0.221
-i	server-instance	<blank>	Instance name provided during database installation.	-i SQLEXPRESS
-p	port	1433	Port of SQL Server	-p 1433
-a	database-admin-user-name	sa	Admin user name that has appropriate rights.	-a sa
-l	login-name	twadmin	Name of login created above manually.	-l twadmin
-d	thingworx-database-name	thingworx	Name of database	-d thingworx
-u	thingworx-user-name	<same-as-login-name>	Name of user that will be created inside the database.	-u twadmin
-s	schema-name	twschema	Name of schema created inside thingworx database.	-s twschema

Configuring and Executing the Model/Data Provider Schema Script (Windows)

1. In the command prompt, execute the below bat file (**thingworxMssqlSchemaSetup.bat**) with the appropriate parameters (listed in the table below):

```
thingworxMssqlSchemaSetup.bat -h <server> -i <server-instance> -p
<port> -l <login-name> -d <thingworx-database-name> -o <option
(all,model,data,property,modelwithproperty)>
```

Upon execution of this bat file, it will ask for password of login (manually created above) for each script. Enter the password and click enter each time.

Upon execution of these scripts, all the tables, indexes, procedures, etc. required for setting up the ThingWorx platform will be created inside the **thingworx** database with default schema **twschema**.

Description of parameters:

Option	Parameter	Default	Description	Example
-h	server	localhost	FQDN or IP of database server	-h 10.0.0.221
-i	server-instance	<blank>	Instance name provided during database installation	-i SQLEXPRESS
-p	port	1433	Port of SQL Server	-p 1433
-l	login-name	twadmin	Name of login created above manually.	-l twadmin
-d	thingworx-database-name	thingworx	Name of database	-d thingworx
-o	option	all	To execute all scripts / model / data / property / model with property will execute.	-o all

Configuring and Executing the Database Cleanup Script (Windows)

The database cleanup bat script (**thingworxMssqlDBCleanup.bat***)* is provided for convenience and for development/testing purposes. Performing the steps in this section is only required if the entire database object and data needs to be wiped out so that you can start from scratch.

1. In the command prompt, execute the below bat file with the appropriate parameters (listed in the table below):

```
thingworxMssqlDBCleanup.bat -h <server> -i <server-instance> -p
<port> -a <database-admin-user-name> -d <thingworx-database-name>
```

Description of parameters:

Option	Parameter	Default	Description	Example
-h	server	localhost	FQDN or IP of database server	-h 10.0.0.221

-i	server-instance	<blank>	Instance name provided during database installation	-i SQLEXPRESS
-p	port	1433	Port of SQL Server	-p 1433
-a	database-admin-user-name	sa	Admin user name which is having appropriate rights.	-a sa
-d	thingworx-database-name	thingworx	Name of database	-d thingworx

Database Setup (Linux)

Before performing the following steps, verify you have performed the steps in the following sections in this document: Installing the ODBC Driver for SQL Server on Linux and Connecting with the Database Server (Linux).

Creating login **twadmin**

1. Open command prompt in Linux client.
2. Use the following commands to create a login on the SQL Server:

```
sqlcmd –s <database-server-name> -u <sql-administrator-username>
-p <password>
CREATE LOGIN twadmin WITH PASSWORD = '<password> '
GO
exit
```

Configuring and Executing the Database Setup Script (Linux)

1. Obtain and copy the ThingWorx software download folder. This folder contains the scripts described in this section.
   NOTE: Software downloads are available on the PTC eSupport page.
2. Open the command prompt and change the directory to script folder.
3. Enter the following command:

```
thingworxMssqlDBSetup.sh -h <server> -i <server-instance> -p
<port> -a <database-admin-user-name> -r <password> -l <login-
name> -d <thingworx-database-name> -u <thingworx-user-name> -s
<schema-name>
```

   Upon execution, it will ask for password of database-admin-user. Enter the password and click enter. The following tasks will be completed by this bat script:

Once the script is executed, a database will be created with a user inside the database and a login associated with the user. This user will be assigned the db_owner role on created database. A schema also will be created if provided on the command line. Authorization of this schema will be given to the created user.

Description of parameters:

Option	Parameter	Default	Description	Example
-h	server	localhost	FQDN or IP of database server	-h 10.0.0.221
-i	server-instance	<blank>	Instance name provided during database installation	-i SQLEXPRESS
-p	port	1433	Port of SQL Server	-p 1433
-a	database-admin-user-name	sa	Admin user name which is having appropriate rights.	-a sa
-r	password	Password@123	Password of database-admin-user	Password@123
-l	login-name	twadmin	Name of login created above manually.	-l twadmin
-d	thingworx-database-name	thingworx	Name of database	-d thingworx
-u	thingworx-user-name	<same-as-login-name>	Name of user which will be created inside the database	-u twadmin
-s	schema-name	twschema	Name of schema created inside thingworx database.	-s twschema

Configuring and Executing the Model/Data Provider Schema Script (Linux)

1. In the command prompt, execute the below shell file with the appropriate parameters:

```
thingworxMssqlSchemaSetup.sh -h <server> -i <server-instance> -p
<port> -l <login-name> -r <password> -d <thingworx-database-name>
-o <option (all,model,data,property,modelwithproperty)>
```

On execution of this shell file, it will ask for password of login (created above manually) for each script. Enter the password and click enter each time.

On execution of these scripts, all the tables, indexes, procedures, etc. required for setting up the ThingWorx Platform will be created inside the **thingworx** database with default schema **twschema**.

Description of parameters:

Option	Parameter	Default	Description	Example
-h	server	localhost	FQDN or IP of database server	-h 10.0.0.221
-i	server-instance	<blank>	Instance name provided during database installation	-i SQLEXPRESS
-p	port	1433	Port of SQL Server	-p 1433
-l	login-name	twadmin	Name of login created above manually.	-l twadmin
-r	password	Password@123	Password of database-admin-user	Password@123
-d	thingworx-database-name	thingworx	Name of database	-d thingworx
-o	option	all	To execute all scripts / model / data / property / model with property will execute.	-o all

Configuring and Executing the Database Cleanup Script (Linux)

The database cleanup shell script is provided for convenience and development/testing purposes. This is only required when the entire database object and data needs to wiped out and start fresh again.

1. In the command prompt, execute the below shell file with the appropriate parameters:

```
thingworxMssqlDBCleanup.sh -h <server> -i <server-instance> -p
<port> -a <database-admin-user-name> -r <password> -d <thingworx-
database-name>
```

Description of parameters:

Option	Parameter	Default	Description	Example
-h	server	localhost	FQDN or IP of database server	-h 10.0.0.221
-i	server-instance	<blank>	Instance name provided during database installation	-i SQLEXPRESS
-p	port	1433	Port of SQL Server	-p 1433
-a	database-admin-user-name	sa	Admin user name which is having appropriate rights.	-a sa
-r	password	Password@123	Password of database-admin-user	Password@123
-d	thingworx-database-name	thingworx	Name of database	-d thingworx

Configuring ThingWorx for MS SQL Server

Refer to the ThingWorx Installation Guide for Java and Tomcat installation and configuration information. The following sections outline the MS SQL -specific configuration instructions that need to be performed in addition to the steps in the Installing Thingworx 7.4 Guide.

Installing the MS SQL JDBC Driver

1. Download the Microsoft JDBC Driver 6.0 for SQL Server driver from the following link: https://www.microsoft.com/en-in/download/details.aspx?id=11774

2. Copy the **sqljdbc42-6.0.*<build_number>*.jar** file to the **lib** directory of the Tomcat installation: `(<TOMCAT_HOME>/lib)`.

3. Restart the Tomcat server to load the JDBC driver to make it available for the ThingWorx web application.

MS SQL Server Configuration Options in ThingWorx

The platform-settings.json file is available in the software download package for administrators to adjust settings for fine-tuning.

1. Create a folder called **ThingworxPlatform** at the root (for example, **\ThingworxPlatform** or as a system variable (for example,

THINGWORX_PLATFORM_SETTINGS=/data/ThingworxPlatform).

2. Place the platform-settings.json file into the **ThingworxPlatform** folder.

3. Configure as necessary. Refer to the configuration options below and the Sample platform-settings.json.

platform-settings.json Options		
Setting	Default	Description
Core Platform Settings		
BackupStorage	/ThingworxBackupStorage	The directory name where all backups are written to.
DatabaseLogRetentionPolicy	7	The number of days that database logs are retained.
EnableBackup	true	Determines whether backups are retained.
EnableHA	false	Determines whether the PlatformThingWorx Core can be configured for a highly available landscape.
EnableSystemLogging	false	Determines whether system logging is enabled. NOTE: DO NOT TURN THIS ON UNLESS INSTRUCTED BY THINGWORX SUPPORT.
HTTPRequestHeaderMaxLength	2000	The maximum allowable length for HTTP Request Headers values.
HTTPRequestParameterMaxLength	2000	The maximum allowable length for HTTP Request Parameter values.
Storage	/ThingworxStorage	The directory where all storage directories are created/located (excluding Backup Storage).
HA Settings Settings specific to a MSSQL HA landscape configuration. All are optional, and are ignored if the EnableHA setting above is set to false.		
CoordinatorConnectionTimeout	10000	How long to wait (in milliseconds) for a connection to be established with process/server used to coordinate ThingWorx leadership.
CoordinatorHosts	Host1:30015;Host2:30015;Host3:30015	In scale-out system a list of host names separated by a semicolon. Only hosts that have the role master should be used. (e.g. "Host1:30015, Host2:30015;Host3:30015").
CoordinatorMaxRetries	3	The maximum allowable number of retries that will be made to establish a connection with the process/server used to coordinate ThingWorx leadership.
CoordinatorRetryTimeout	3000	How long to wait (in milliseconds) for each retry attempt.
CoordinatorSessionTimeout	90000	How long the ThingWorx session is to wait (in milliseconds) without

		receiving a "heartbeat" from the process/server used to coordinate ThingWorx Core leadership.
LoadBalancerBase64EncodedCredentials	QWRtaW5pc3RyYXRvcjphZG1pbg==	The Base64-encoded credentials for the HA Load Balancer, in the format of <user>:<password>.
MssqlPersistenceProviderPackage Contains MSSQL-specific Persistence Provider settings. If MS SQL is not the Persistence Provider, then this entire section should be ignored.		
driverClass	"com.microsoft.sqlserver.jdbc.SQLServerDriver"	The fully-qualified class name of the JDBC driverClass that is expected to provide Connections.
jdbcUrl	"jdbc:sqlserver://localhost:1433;databaseName=thingworx;applicationName=Thingworx;"	The jdbcUrl url used to connect to MSSQL
username	twadmin	This is the userid that owns the TWSCHEMA schema and is used for authentication to MSSQL in the JDBC connection string.
password	Password@123	
acquireIncrement	5	Determines how many connections at a time the ThingWorx will try to acquire when the pool is exhausted.
acquireRetryAttempts	3	Defines how many times ThingWorx will try to acquire a new Connection from the database before giving up.
acquireRetryDelay	10000	The time (in milliseconds) ThingWorx will wait between acquire attempts.
checkoutTimeout	1000000	The number of milliseconds a client calling getConnection() will wait for a Connection to be checked-in or acquired when the pool is exhausted.
fetchSize	5000	The count of rows to be fetched in batches instead of caching all rows on the client side.
idleConnectionTestPeriod	60	If this is a number greater than 0, ThingWorx will test all idle, pooled but unchecked-out connections, every x number of seconds.
initialPoolSize	5	Initial number of database connections created and maintained within a pool upon

		startup. Should be between minPoolSize and maxPoolSize.
maxConnectionAge	0	Seconds, effectively a time to live. A Connection older than maxConnectionAge will be destroyed and purged from the pool.
maxIdleTime	0	Seconds a connection can remain pooled but unused before being discarded. Zero means idle connections never expire.
maxIdleTimeExcessConnections	300	The number of seconds that connections in excess of minPoolSize are permitted to remain in idle in the pool before being culled. Intended for applications that wish to aggressively minimize the number of open connections, shrinking the pool back towards minPoolSize if, following a spike, the load level diminishes and Connections acquired are no longer needed. If maxIdleTime is set, maxIdleTimeExcessConnections should be smaller to have any effect. Setting this to zero means no enforcement and excess connections are not idled out.
maxPoolSize	100	Maximum number of Connections a pool will maintain at any given time.
maxStatements	100	The size of the ThingWorx global PreparedStatement cache.
minPoolSize	5	Minimum number of Connections a pool will maintain at any given time.
testConnectionOnCheckout	false	If true, an operation will be performed at every connection checkout to verify that the connection is valid.

unreturnedConnectionTimeout	0	The number of seconds to wait for a response from an unresponsive connection before discarding it. If set, if an application checks out but then fails to check-in a connection within the specified period of time, the pool will discard the connection. This permits applications with occasional connection leaks to survive, rather than eventually exhausting the Connection pool. Zero means no timeout, and applications are expected to close their own connections.
Stream Processor Settings		
maximumBlockSize	2500	The maximum number of stream writes to process in one block.
maximumQueueSize	250000	The maximum number of stream entries to queue (will be rejected after that)
maximumWaitTime	10000	Number of milliseconds the system waits before flushing the stream buffer.
numberOfProcessingThreads	5	The number of processing threads.
scanRate	5	The buffer status is checked at the specified rate value in milliseconds.
sizeThreshold	1000	Maximum number of items to accumulate before flushing the stream buffer.
Value Stream Processor Settings		
maximumBlockSize	2500	Maximum number of value stream writes to process in one block.
maximumQueueSize	500000	Maximum number of value stream entries to queue (will be rejected after that).
maximumWaitTime	10000	Number of milliseconds the system waits before flushing the value stream buffer.
numberofProcessingThreads	5	The number of processing threads.
scanRate	5	The rate (in milliseconds) before flushing the stream buffer.
sizeThreshold	1000	Maximum number of items to accumulate before flushing the value stream buffer.

Performance Monitoring and Tuning Tools in MS SQL Server

Microsoft SQL Server provides a comprehensive set of tools for monitoring events in SQL Server and for tuning the physical database design. The choice of tool depends on the type of monitoring or tuning to be done and the particular events to be monitored.

Refer to the following link for more information:
https://msdn.microsoft.com/en-us/library/ms179428.aspx

Microsoft SQL Server Native Backup and Restore Support

A copy of SQL Server data that can be used to restore and recover the data after a failure. A backup of SQL Server data is created at the level of a database or one or more of its files or filegroups. Table-level backups cannot be created. In addition to data backups, the full recovery model requires creating backups of the transaction log.

recovery model
A database property that controls transaction log maintenance on a database. Three recovery models exist: simple, full, and bulk-logged. The recovery model of database determines its backup and restore requirements.

restore
A multi-phase process that copies all the data and log pages from a specified SQL Server backup to a specified database, and then rolls forward all the transactions that are logged in the backup by applying logged changes to bring the data forward in time.

For more information, see https://msdn.microsoft.com/en-in/library/ms187510(v=sql.120).aspx
https://msdn.microsoft.com/en-us/library/ms177429(v=sql.120).aspx

Scaling Out SQL Server

Scalability is the ability of an application to efficiently use more resources in order to do more useful work.

Scalable Shared Databases

The easiest scale out solution to implement in SQL Server is Scalable Shared Databases. In this scenario, you create a database on a SAN, and up to eight SQL Server instances running on different servers attach to the database, and start handling queries. This is the classic "shared disk"–style scale out solution, where processing power is scaled out, but only a single disk image of the data is used. At this point, those that are familiar with SQL Server might have questions such as: "But what happens to the locks? I thought each SQL Server instance kept its own locks in its own memory." This is true. Each instance will maintain its own database locks, and none of the instances will know about the other instances' locks. The only way this will work is if there are no locks, and thus Scalable Shared Databases work only if the database is attached as a Read Only database. This means that Scalable Shared Databases are great for data warehouses or reporting databases, but they are not suitable for applications that update data. Going back to our data characteristics, Scalable Shared Databases work only if the Update Frequency is zero. This data is, by definition, historical, and therefore it is all reference data. Figure 1 illustrates the use of Scalable Shared Databases as a scale out solution.

Index Size Limitation and Implementation

In MS SQL Server, the maximum number of bytes in any index key cannot exceed 900 bytes. Though a key can be defined using variable-length columns whose maximum sizes add up to more than 900, but in that case no row must not be inserted with more than 900 bytes of data in those columns. (https://msdn.microsoft.com/en-in/library/ms143432(v=sql.120).aspx).

NOTE: ThingWorx users should be mindful of creating composite keys and their corresponding length. Users should design their key names to be short yet descriptive as possible.

Appendix: Sample platform-settings. json


```
{
   "PlatformSettingsConfig": {
      "BasicSettings": {
         "BackupStorage": "/twdata-hana/ThingworxBackupStorage",
         "DatabaseLogRetentionPolicy": 7,
         "EnableBackup": true,
         "EnableHA": false,
         "EnableSystemLogging": true,
         "HTTPRequestHeaderMaxLength": 2000,
         "HTTPRequestParameterMaxLength": 2000,
         "Storage": "/twdata-hana/ThingworxStorage"
      },

                  "HASettings": {
                        "CoordinatorConnectionTimeout": 10000,
                        "CoordinatorHosts": "127.0.0.1:2181",
                        "CoordinatorMaxRetries": 3,
                        "CoordinatorRetryTimeout": 3000,
                        "CoordinatorSessionTimeout": 90000,
                        "LoadBalancerBase64EncodedCredentials":
"QWRtaW5pc3RyYXRvcjphZG1pbg=="
                  }
          },

                  "MssqlPersistenceProviderPackage": {
                        "ConnectionInformation": {
                              "acquireIncrement": 5,
                              "acquireRetryAttempts": 3,
```

```
                "acquireRetryDelay": 10000,
                "checkoutTimeout": 1000000,
                "driverClass": "com.microsoft.sqlserver.jdbc.SQLServerDriver",
                "fetchSize": 5000,
                "idleConnectionTestPeriod": 60,
                "initialPoolSize": 5,
                "jdbcUrl":
"jdbc:sqlserver://localhost:1433;databaseName=thingworx;applicationName=Thingworx;",
                "maxConnectionAge": 0,
                "maxIdleTime": 0,
                "maxIdleTimeExcessConnections": 300,
                "maxPoolSize": 100,
                "maxStatements": 100,
                "minPoolSize": 5,
                "numHelperThreads": 8,
                "password": "Password@123",
                "testConnectionOnCheckout": false,
                "unreturnedConnectionTimeout": 0,
                "username": "msadmin",
        },
        "StreamProcessorSettings": {
                "maximumBlockSize": 2500,
                "maximumQueueSize": 250000,
                "maximumWaitTime": 10000,
                "numberOfProcessingThreads": 5,
                "scanRate": 5,
                "sizeThreshold": 1000
        },
```

```
                "ValueStreamProcessorSettings": {
                    "maximumBlockSize": 2500,
                    "maximumWaitTime": 10000,
                    "maximumQueueSize": 500000,
                    "numberOfProcessingThreads": 5,
                    "scanRate": 5,
                    "sizeThreshold": 1000
                }
            }

        }
}
```