Progress DataDirect for JDBC for HubSpot

User's Guide

Release 6.0.0

Copyright

Table of Contents

1

Welcome to the Progress DataDirect for JDBC for HubSpot Driver

The Progress® DataDirect® for JDBC™ for HubSpot™ driver supports SQL read-only access for JDBC applications to HubSpot. To support SQL access to HubSpot, the driver creates a relational map of the HubSpot data model and translates SQL statements to HubSpot requests. In addition, the driver employs a SQL engine component that provides support to SQL constructs unavailable in HubSpot. This functionality offers a number of advantages, including support for reporting data and metadata in a form that JDBC applications are ready to use.

For an overview of how the driver exposes the HubSpot data model as a relational schema, and how to query HubSpot, see Introduction to the HubSpot Data Model.

The documentation for the driver also includes the *Progress DataDirect for JDBC Drivers Reference*. The reference provides general reference information for all DataDirect drivers for JDBC, including content on troubleshooting, supported SQL escapes, and DataDirect tools.

For the complete documentation set, visit the Progress DataDirect Connectors Documentation Hub: https://docs.progress.com/category/datadirect-hubspot.

For details, see the following topics:

- What's new in this release?
- Requirements
- Installing and setting up the driver
- Driver and DataSource classes
- Connection URL examples
- Data types

- SQL escape sequences
- DataDirect tools
- Troubleshooting
- Contacting Technical Support

What's new in this release?

Support and certification

Visit the following web pages for the latest support and certification information.

- Release Notes: https://www.progress.com/datadirect-connectors/whats-new#jdbc
- DataDirect Product Compatibility Guide: https://docs.progress.com/bundle/datadirect-product-compatibility/resource/datadirect-product-compatibility.pdf

Highlights of 6.0.0 Release

- The driver supports SQL read-only access to HubSpot. See Supported SQL statements and extensions on page 61 for details.
- The driver supports JDBC core functions. For details, refer to "JDBC support" in the *Progress DataDirect for JDBC Drivers Reference*.
- The driver supports HubSpot data types through data type inference. See Data types on page 16 and getTypeInfo() on page 17 for details.
- The driver supports OAuth 2.0 authentication. See OAuth 2.0 authentication on page 42 for supported grant types and further details.
- The driver supports API Key authentication. See API key authentication on page 44 for details.
- The driver supports the handling of large result sets with paging and the FetchSize on page 53 connection property.

Requirements

The driver is compatible with JDBC 2.0, 3.0, and 4.0.

The driver requires a Java Virtual Machine (JVM) that is Java SE 8 or higher, including Oracle JDK, OpenJDK, and IBM SDK (Java) distributions.

Installing and setting up the driver

This section provides you with an overview of the steps required to install and set-up the driver. After completing this procedure, you will be able to begin accessing data with your application.

To begin accessing data with the driver:

1. Install the driver:

   a) After downloading the product, unzip the installer files to a temporary directory.

   b) From the installer directory, run the appropriate installer file to start the installer.

   - **Windows**: `PROGRESS_DATADIRECT_JDBC_INSTALL.exe`
   - **Non-Windows**: `PROGRESS_DATADIRECT_JDBC_INSTALL.jar`

   c) Follow the prompts to complete installation.

   The installer program supports multiple installation methods, including command-line and silent installations. For detailed instructions, refer to the *Progress DataDirect for JDBC Drivers Installation Guide*.

2. Set your system CLASSPATH to include the driver `.jar` file. The CLASSPATH is the search string your Java Virtual Machine (JVM) uses to locate JDBC drivers on your computer. The following examples demonstrate setting the CLASSPATH from a command line using the default installation directory.

   - **Windows Example**

     ```
     CLASSPATH=.;C:\Program Files\Progress\DataDirect\JDBC\lib\60\hubspot.jar
     ```

   - **UNIX/LINUX Example**

     ```
     CLASSPATH=.:/opt/Progress/DataDirect/JDBC/lib/60/hubspot.jar
     ```

3. Configure your driver using one of the following methods:

   - **Connection URL**: You can begin using the driver immediately by passing a connection URL with your application or tool. The following examples show how to connect using OAuth 2.0 access token, OAuth 2.0 refresh token, or API key authentication.

     OAuth 2.0 access token

     ```
     jdbc:datadirect:hubspot:AuthenticationMethod=OAuth2;7
     AccessToken=abcDEF123ghi-456Jklmno789-Pqrst01234;
     ```

     OAuth 2.0 refresh token

     ```
     jdbc:datadirect:hubspot:AuthenticationMethod=oauth2;
     ClientID=ab123c45-def6-7g89-gh1i-m2345no67891;
     ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd;
     RefreshToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;
     ```

     Note: See OAuth 2.0 authentication on page 42 for details.

     API key

     ```
     jdbc:datadirect:hubspot:AuthenticationMethod=URLParameter;
     APIKey=1234ab5cd-1a2b-3a4b-678uvw-xyz91011;
     ```

     Note: See API key authentication on page 44 for details.

   - **Data sources**: The driver also supports connecting using JDBC data sources. A JDBC data source is a Java object, specifically a DataSource object, that defines connection information required for a JDBC driver to connect to the database. See Connecting using data sources for more information.

Note: For most connections, specifying the minimum required connection properties is sufficient to begin accessing data; however, you can provide values for optional properties to use additional supported features and improve performance.

4. Set the values for any optional properties that you want to configure. For additional information on optional features and functionality, see the following resources:
   - Connection URL Examples provides connection string examples that can be used to configure common functionality and features. You can modify and combine these examples to create a string that best suites your environment.
   - Connection property descriptions provides a complete list of supported properties by functionality.
   - Performance considerations describes connection properties that affect performance, along with recommended settings.
5. Connect to your service and begin accessing data with your applications, BI tools, database tools, and more. To help you get started, the following resources guide you through accessing data with some common tools:
   - Tableau: Tableau is a business intelligence software program that allows you to easily create reports and visualized representations of your data.
   - DbVisualizer: DB Visualizer is a database tool that allows you to connect and execute SQL statements against your data.
   - DataDirect Test: DataDirect Test allows you to test connect, execute SQL statements, and practice using the JDBC API right out of the box.
   - Supported SQL statements and extensions: This section describes the syntax used for SQL statements supported by the driver. You can modify and use the provided examples for your application or tool.

This completes the deployment of the driver.

Driver and DataSource classes

The following are the `Driver` and `DataSource` classes used by the driver:

Driver class:

com.ddtek.jdbc.hubspot.HubSpotDriver

DataSource class:

com.ddtek.jdbcx.hubspot.HubSpotDataSource

Connection URL examples

After setting the CLASSPATH, the connection information needs to be passed in the form of a connection URL. This section provides examples of connection strings configured to use common features and functionality. You can modify and/or combine these examples to create a connection string for your environment.

Note:

- You can also use the DataDirect Configuration Manager tool to generate and test connection URLs. For more information, see "Generating connection URLs with the Configuration Manager."
- Connection property names are case-insensitive. For example, `Password` is the same as `password`.
- For connection properties that support string values, use the following escape sequence to specify values containing leading or trailing spaces and curly brackets: `{value}`. For example: `User={hello }` or `Password={{hello}}`.

OAuth 2.0 access token flow

Typically, an OAuth 2.0 grant type or access flow is handled by the application. However, in some scenarios, you may need to secure an access token using external processes. In those instances, you can use the access token flow to access the service manually.

Note: Access tokens are temporary and must be replaced to maintain the session without interruption. The life of an access token is typically six hours.

The following string includes the properties used to connect with the OAuth 2.0 access token flow.

```
jdbc:datadirect:hubspot:AuthenticationMethod=OAuth2;
AccessToken=access_token;[property=value[;...]];
```

where:

access_token

specifies the access token required to authenticate to HubSpot. This property allows you to set the access token manually.

property=value

specifies connection property settings. Multiple properties are separated by a semi-colon.

The following example connection string includes the properties required for connecting with the OAuth 2.0 access token flow.

```
Connection conn = DriverManager.getConnection
  ("jdbc:datadirect:hubspot:AuthenticationMethod=OAuth2;
    AccessToken=abcDEF123ghi-456Jklmno789-Pqrst01234;");
```

See also

Connection property descriptions on page 45
Access token flow on page 42

OAuth 2.0 refresh token grant

This string includes the properties used to connect with OAuth 2.0 refresh token.

```
jdbc:datadirect:hubspot:AuthenticationMethod=OAuth2;
ClientID=client_id;ClientSecret=client_secret;
RefreshToken=refresh_token;Scope=scope;[property=value[;...]];
```

where:

client_id

specifies the client ID key for your application when authenticating with OAuth 2.0.

client_secret

specifies the client secret for your application when authenticating with OAuth 2.0.

Important: The client secret is a confidential value used to authenticate the application to the server. To prevent unauthorized access, this value must be securely maintained.

refresh_token

specifies the refresh token used to either request a new access token or renew an expired access token.

Important: The refresh token is a confidential value used to authenticate to the server. To prevent unauthorized access, this value must be securely maintained.

scope

(optional) specifies a space-separated list of OAuth scopes that limit the permissions granted by an access token.

property=value

specifies connection property settings. Multiple properties are separated by a semi-colon.

The following example connection string includes the properties required for connecting with the OAuth 2.0 refresh token grant.

```
Connection conn = DriverManager.getConnection
  ("jdbc:datadirect:hubspot:AuthenticationMethod=oauth2;
    ClientID=ab123c45-def6-7g89-gh1i-m2345no67891;
    ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd;
    RefreshToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;");
```

See also

Connection property descriptions on page 45
Refresh token grant on page 43

API key

This string includes the properties used to connect with API key authentication.

```
jdbc:datadirect:hubspot:AuthenticationMethod=URLParameter;
APIKey=api_key;[property=value[;...]];
```

where:

api_key

specifies the API key when authenticating with API key authentication.

property=value

specifies connection property settings. Multiple properties are separated by a semi-colon.

The following example connection string includes the properties required for connecting with the API key authentication.

```
Connection conn = DriverManager.getConnection
  ("jdbc:datadirect:hubspot:AuthenticationMethod=URLParameter;
    APIKey=1234ab5cd-1a2b-3a4b-678uvw-xyz91011;");
```

See also

Connection property descriptions on page 45
API key authentication on page 44

Proxy server

This string includes the properties you may need to connect through a proxy server with OAuth 2.0 refresh token grant authentication.

```
jdbc:datadirect:hubspot:ProxyHost=proxy_host;ProxyPassword=proxy_password;
ProxyPort=proxy_port;ProxyUser=proxy_user;AuthenticationMethod=OAuth2;
ClientID=client_id;ClientSecret=client_secret;
RefreshToken=refresh_token;[property=value[;...]];
```

where:

proxy_host

specifies the proxy server to use for the first connection.

proxy_password

specifies the password needed to connect to a proxy server for the first connection.

proxy_port

specifies the port number where the proxy server is listening for requests for the first connection. The default is `0`.

proxy_user

specifies the user name needed to connect to a proxy server for the first connection.

client_id

specifies the client ID key for your application when authenticating with OAuth 2.0.

client_secret

specifies the client secret for your application when authenticating with OAuth 2.0.

Important: The client secret is a confidential value used to authenticate the application to the server. To prevent unauthorized access, this value must be securely maintained.

refresh_token

specifies the refresh token used to either request a new access token or renew an expired access token.

property=value

specifies connection property settings. Multiple properties are separated by a semi-colon.

The following example connection string includes the properties required for using a proxy server with OAuth 2.0 refresh token grant authentication.

```
Connection conn = DriverManager.getConnection
  ("jdbc:datadirect:hubspot:
    ProxyHost=pserver;ProxyPassword=secret;ProxyPort=808;ProxyUser=jsmith;
    AuthenticationMethod=OAuth2;ClientID=ab123c45-def6-7g89-gh1i-m2345no67891;
    ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd;
    RefreshToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;");
```

See also

Connection property descriptions on page 45

Data types

The following table lists data types supported by the driver and how they are mapped to JDBC data types.

See "getTypeInfo()" for getTypeInfo() results of data types supported by the driver.

Table 1: HubSpot Data Types

HubSpot Data Type	JDBC Data Type
BigInt	BIGINT
Binary	BINARY
Bit	BIT
Boolean	BOOLEAN

HubSpot Data Type	JDBC Data Type
Char	CHAR
Date	DATE
Decimal	DECIMAL
Double	DOUBLE
Float	FLOAT
GUID	CHAR
Integer	INTEGER
JSON	VARCHAR
LongVarBinary	LONGVARBINARY
LongVarChar	LONGVARCHAR
NVarChar	NVARCHAR
SmallInt	SMALLINT
Time	TIME
Timestamp	TIMESTAMP
TinyInt	TINYINT
VarBinary	VARBINARY
VarChar	VARCHAR

getTypeInfo()

The DatabaseMetaData.getTypeInfo() method returns information about data types. The following table provides getTypeInfo() results for supported data types.

Table 2: getTypeInfo() Results

TYPE_NAME = BigInt AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = -5 (BIGINT) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = BigInt MAXIMUM_SCALE = 0	MINIMUM_SCALE = 0 NULLABLE = 1 NUM_PREC_RADIX = 10 PRECISION = 19 SEARCHABLE = 3 SQL_DATA_TYPE = 25 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = FALSE
TYPE_NAME = Binary AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = -2 (BINARY) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = X' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = Binary MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 32767 SEARCHABLE = 3 SQL_DATA_TYPE = 60 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = Bit AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = -7 (BIT) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = X' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = Bit MAXIMUM_SCALE = 0	MINIMUM_SCALE = 0 NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 1 SEARCHABLE = 3 SQL_DATA_TYPE = 14 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL

TYPE_NAME = Boolean AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 16 (BOOLEAN) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = Boolean MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 1 SEARCHABLE = 3 SQL_DATA_TYPE = 16 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = Char AUTO_INCREMENT = FALSE CASE_SENSITIVE = TRUE CREATE_PARAMS = NULL DATA_TYPE = 1 (CHAR) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = Char MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 255 SEARCHABLE = 3 SQL_DATA_TYPE = 1 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = Date AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 91 (DATE) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = DATE ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = Date MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 10 SEARCHABLE = 3 SQL_DATA_TYPE = 91 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL

TYPE_NAME = Decimal AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 3 (DECIMAL) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = Decimal MAXIMUM_SCALE = 1000	MINIMUM_SCALE = 0 NULLABLE = 1 NUM_PREC_RADIX = 10 PRECISION = 1000 SEARCHABLE = 3 SQL_DATA_TYPE = 3 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = FALSE
TYPE_NAME = Double AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 8 (DOUBLE) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = Double MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = 2 PRECISION = 53 SEARCHABLE = 3 SQL_DATA_TYPE = 8 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = FALSE
TYPE_NAME = Float AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 6 (FLOAT) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = Float MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = 2 PRECISION = 24 SEARCHABLE = 3 SQL_DATA_TYPE = 6 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = FALSE

TYPE_NAME = GUID AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 1 (CHAR) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = GUID MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 36 SEARCHABLE = 3 SQL_DATA_TYPE = 1 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = Integer AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 4 (INTEGER) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = Integer MAXIMUM_SCALE = 0	MINIMUM_SCALE = 0 NULLABLE = 1 NUM_PREC_RADIX = 10 PRECISION = 10 SEARCHABLE = 3 SQL_DATA_TYPE = 4 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = FALSE
TYPE_NAME = JSON AUTO_INCREMENT = FALSE CASE_SENSITIVE = TRUE CREATE_PARAMS = NULL DATA_TYPE = 12 (VARCHAR) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = JSON MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 16777215 SEARCHABLE = 3 SQL_DATA_TYPE = 12 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL

TYPE_NAME = LongVarBinary AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = -4 (LONGVARBINARY) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = X' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = LongVarBinary MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 16777215 SEARCHABLE = 0 SQL_DATA_TYPE = -4 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = LongVarChar AUTO_INCREMENT = FALSE CASE_SENSITIVE = TRUE CREATE_PARAMS = NULL DATA_TYPE = -1 (LONGVARCHAR) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = LongVarChar MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 16777215 SEARCHABLE = 0 SQL_DATA_TYPE = -1 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = NVarChar AUTO_INCREMENT = FALSE CASE_SENSITIVE = TRUE CREATE_PARAMS = NULL DATA_TYPE = -9 (NVARCHAR) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = NVarChar MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 32767 SEARCHABLE = 3 SQL_DATA_TYPE = -9 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL

TYPE_NAME = SmallInt AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 5 (SMALLINT) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = SmallInt MAXIMUM_SCALE = 0	MINIMUM_SCALE = 0 NULLABLE = 1 NUM_PREC_RADIX = 10 PRECISION = 5 SEARCHABLE = 3 SQL_DATA_TYPE = 5 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = FALSE
TYPE_NAME = Time AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 92 (TIME) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = TIME ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = Time MAXIMUM_SCALE = 9	MINIMUM_SCALE = 0 NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 12 SEARCHABLE = 3 SQL_DATA_TYPE = 92 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = Timestamp AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = 93 (TIMESTAMP) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = TIMESTAMP ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = Timestamp MAXIMUM_SCALE = 9	MINIMUM_SCALE = 0 NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 23 SEARCHABLE = 3 SQL_DATA_TYPE = 93 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL

TYPE_NAME = TinyInt AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = -6 (TINYINT) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = NULL LITERAL_SUFFIX = NULL LOCAL_TYPE_NAME = TinyInt MAXIMUM_SCALE = 0	MINIMUM_SCALE = 0 NULLABLE = 1 NUM_PREC_RADIX = 10 PRECISION = 3 SEARCHABLE = 3 SQL_DATA_TYPE = -6 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = FALSE
TYPE_NAME = VarBinary AUTO_INCREMENT = FALSE CASE_SENSITIVE = FALSE CREATE_PARAMS = NULL DATA_TYPE = -3 (VARBINARY) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = X' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = VarBinary MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 16777215 SEARCHABLE = 3 SQL_DATA_TYPE = 61 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL
TYPE_NAME = VarChar AUTO_INCREMENT = FALSE CASE_SENSITIVE = TRUE CREATE_PARAMS = NULL DATA_TYPE = 12 (VARCHAR) FIXED_PREC_SCALE = FALSE LITERAL_PREFIX = ' LITERAL_SUFFIX = ' LOCAL_TYPE_NAME = VarChar MAXIMUM_SCALE = NULL	MINIMUM_SCALE = NULL NULLABLE = 1 NUM_PREC_RADIX = NULL PRECISION = 32767 SEARCHABLE = 3 SQL_DATA_TYPE = 12 SQL_DATETIME_SUB = NULL UNSIGNED_ATTRIBUTE = NULL

SQL escape sequences

The driver supports the following SQL escape sequences.

- Date, Time, and Timestamp Escape Sequences
- Scalar Functions
- Outer Join Escape Sequences
- LIKE Escape Character Sequence for Wildcards

Refer to "SQL escape sequences" in the *Progress DataDirect for JDBC Drivers Reference* for information about SQL escape sequences.

Supported scalar functions

The driver supports the scalar functions in the following table. Note that your database system may not support all these functions. Refer to the documentation for your database system to find out which functions are supported by your database.

In addition, you can also determine the supported scalar functions by using DatabaseMetaData methods.

You can use scalar functions in SQL statements with the following syntax:

`{fn scalar-function}`

where:

scalar-function

is a scalar function supported by the drivers, as listed in the following table.

Example:

```
SELECT id, name FROM emp WHERE name LIKE {fn UCASE('Smith')}
```

Refer to "Scalar functions" in the *Progress DataDirect for JDBC Drivers Reference* for more information.

Table 3: Supported Scalar Functions

String Functions	Numeric Functions	Timedate Functions	System Functions
ASCII	ABS	CURDATE	CURSESSIONID
BIT_LENGTH	ACOS	CURRENT_DATE	DATABASE
CHAR	ASIN	CURRENT_TIME	IDENTITY
CHAR_LENGTH	ATAN	CURRENT_TIMESTAMP	USER
CHARACTER_LENGTH	ATAN2	CURTIME	
CONCAT	BITAND	DATEDIFF	

String Functions	Numeric Functions	Timedate Functions	System Functions
DIFFERENCE	BITOR	DATE_ADD	
HEXTORAW	BITXOR	DATE_SUB	
INSERT	CEILING	DAY	
LCASE	COS	DAYNAME	
LEFT	COT	DAYOFMONTH	
LENGTH	DEGREES	DAYOFWEEK	
LOCATE	EXP	DAYOFYEAR	
LOCATE_2	FLOOR	EXTRACT	
LOWER	LOG	HOUR	
LTRIM	LOG10	MINUTE	
OCTET_LENGTH	MOD	MONTH	
RAWTOHEX	PI	MONTHNAME	
REPEAT	POWER	NOW	
REPLACE	RADIANS	QUARTER	
RIGHT	RAND	SECOND	
RTRIM	ROUND	SECONDS_SINCE_MIDNIGHT	
SOUNDEX	ROUNDMAGIC	TIMESTAMPADD	
SPACE	SIGN	TIMESTAMPDIFF	
SUBSTR	SIN	TO_CHAR	
SUBSTRING	SQRT	WEEK	
UCASE	TAN	YEAR	
UPPER	TRUNCATE		

DataDirect tools

Progress DataDirect for JDBC drivers install the set of tools described in this section. For detailed instructions on using these tools, refer to the corresponding topics in the *Progress DataDirect for JDBC Drivers Reference*.

- DataDirect Test allows you to test your JDBC driver and learn the JDBC API.
- DataDirect Connection Pool Manager allows you to pool connections when accessing databases. When your applications use connection pooling, connections are reused rather than created each time a connection is requested. Because establishing a connection is among the most costly operations an application may perform, using Connection Pool Manager to implement connection pooling can significantly improve performance.
- Statement Pool Monitor loads statements into and remove statements from the statement pool as well as generate information to help you troubleshoot statement pooling performance.
- DataDirect Spy logs detailed information about calls your driver makes that can be used for troubleshooting.

Troubleshooting

The *Progress DataDirect for JDBC Drivers Reference* provides information on troubleshooting problems should they occur. Refer to the "Troubleshooting" section in the *Reference* for details.

Contacting Technical Support

Progress DataDirect offers a variety of options to meet your support needs. Please visit our Web site for more details and for contact information:

https://www.progress.com/support

The Progress DataDirect Web site provides the latest support information through our global service network. The SupportLink program provides access to support contact details, tools, patches, and valuable information, including a list of FAQs for each product. In addition, you can search our Knowledgebase for technical bulletins and other information.

When you contact us for assistance, please provide the following information:

- Your number or the serial number that corresponds to the product for which you are seeking support, or a case number if you have been provided one for your issue. If you do not have a SupportLink contract, the SupportLink representative assisting you will connect you with our Sales team.
- Your name, phone number, email address, and organization. For a first-time call, you may be asked for full information, including location.
- The Progress DataDirect product and the version that you are using.
- The type and version of the operating system where you have installed your product.
- Any database, database version, third-party software, or other environment information required to understand the problem.
- A brief description of the problem, including, but not limited to, any error messages you have received, what steps you followed prior to the initial occurrence of the problem, any trace logs capturing the issue, and so

on. Depending on the complexity of the problem, you may be asked to submit an example or reproducible application so that the issue can be re-created.

- A description of what you have attempted to resolve the issue. If you have researched your issue on Web search engines, our Knowledgebase, or have tested additional configurations, applications, or other vendor products, you will want to carefully note everything you have already attempted.
- A simple assessment of how the severity of the issue is impacting your organization.

2

Tutorials

The following sections guide you through using the driver to access your data with some common third-party applications. For information on installing your driver and setting the CLASSPATH, see "Installing and setting-up the driver."

For details, see the following topics:

- Tableau
- DbVisualizer

Tableau

After you have installed your driver and defined it on the CLASSPATH, you can use the driver to access your data with Tableau. Tableau is a business intelligence software program that allows you to easily create reports and visualized representations of your data. By using the driver with Tableau, you can improve performance when retrieving data while leveraging the driver's relational mapping tools.

To use the driver to access data with Tableau:

1. Navigate to the `\lib\xx` subdirectory of the Progress DataDirect installation directory; then, locate the `jar` file for your driver:

   ```
   hubspot.jar
   ```

2. Copy the `.jar` file for your driver into the following directory:

   Windows: `C:\Program Files\Tableau\Drivers`

Linux: `/opt/tableau/tableau_driver/jdbc`

3. Open Tableau. From the **Connect** menu, select **Other Databases (JDBC)**.
4. In the **Other Databases (JDBC)** dialog, provide values for the following fields; then, click **Sign In**.
   - **URL**: Copy and paste your connection URL into this field. The following examples show how to connect using OAuth 2.0 access token, OAuth 2.0 refresh token, or API key authentication.

     OAuth 2.0 access token

     ```
     jdbc:datadirect:hubspot:AuthenticationMethod=OAuth2;
     AccessToken=abcDEF123ghi-456Jklmno789-Pqrst01234;
     ```

     OAuth 2.0 refresh token

     ```
     jdbc:datadirect:hubspot:AuthenticationMethod=oauth2;
     ClientID=ab123c45-def6-7g89-gh1i-m2345no67891;
     ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd;
     RefreshToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;
     ```

     Note: See OAuth 2.0 authentication on page 42 for details.

     API key

     ```
     jdbc:datadirect:hubspot:AuthenticationMethod=URLParameter;
     APIKey=1234ab5cd-1a2b-3a4b-678uvw-xyz91011;
     ```

     Note: See API key authentication on page 44 for details.

   - **Dialect**: Select **SQL92** (the default) from the drop-down box.
   - **Username**: If required by the authentication method being used, enter the user name. Alternatively, this value can be specified with the `User` property in the connection string.
   - **Password**: If required by the authentication method being used, enter the password. Alternatively, this value can be specified with the `Password` property in the connection string.
5. The **Data Source** window appears. In the **Schema** field, select the schema for the service you want to use.
6. In the **Table** field, the tables stored in the selected schema are now exposed and available for selection.

You have successfully accessed your data and are now ready to create reports with Tableau. For detailed information, refer to the Tableau product documentation at: https://www.tableau.com/support/help.

DbVisualizer

After you have installed your driver and defined it on the CLASSPATH, you can use the driver to access your data with the third-party DbVisualizer tool. The following topics guide you through using DbVisualizer to add your driver, connect, and execute SQL statements.

Adding a driver

To add a driver with DbVisualizer:

1. Open DbVisualizer.
2. From the menu, select **Tools**>**Driver Manager**. The Driver Manager window opens.
3. From the Driver Manager menu, select **Driver**>**Create Driver**.
4. Click the  button to navigate to the location of the driver jar file; then, click **OK**. The following are the default locations for the driver:

   Windows

   ```
   C:\Program Files\Progress\DataDirect\JDBC\lib\60\hubspot.jar
   ```

   Linux

   ```
   /opt/Progress/DataDirect/JDBC/lib/60/hubspot.jar
   ```

5. Provide values for the following fields; then, close the Driver Manager window.
   - **Name**: Type an alias for your driver. For example:

     ```
     HubSpot
     ```

   - **URL Format**: Optionally, specify the format of the connection string for your driver. For example:

     ```
     jdbc:datadirect:hubspot:
     ```

   - **Driver Class**: From the drop down menu, select the driver class for your driver. For example:

     ```
     com.ddtek.jdbcx.hubspot.HubSpotDataSource
     ```

You can now use your driver with DbVisualizer. Proceed to "Connecting and executing SQL statements" for information on connecting and executing SQL statements.

Connecting and executing SQL statements

To use the driver to access data with DbVisualizer:

1. Open DbVisualizer.
2. From the menu, select **Database**>**New Connection**. When prompted to use the Connection Wizard, click **OK**.
3. Provide the following information when prompted; then, click **Next** to proceed:
   - **Connection alias**: Type the name to be used when referring to this connection.
   - **Driver**: Select the alias that you provided for your driver from the drop-down menu.
4. Provide values for the following fields; then, click **Finish**.
   - **Database URL**: Copy and paste your connection URL into this field. The following examples show how to connect using OAuth 2.0 access token, OAuth 2.0 refresh token, or API key authentication.

     OAuth 2.0 access token

     ```
     jdbc:datadirect:hubspot:AuthenticationMethod=OAuth2;
     AccessToken=abcDEF123ghi-456Jklmno789-Pqrst01234;
     ```

OAuth 2.0 refresh token

```
jdbc:datadirect:hubspot:AuthenticationMethod=oauth2;
ClientID=ab123c45-def6-7g89-gh1i-m2345no67891;
ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd;
RefreshToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;
```

Note: See OAuth 2.0 authentication on page 42 for details.

API key

```
jdbc:datadirect:hubspot:AuthenticationMethod=URLParameter;
APIKey=1234ab5cd-1a2b-3a4b-678uvw-xyz91011;
```

Note: See API key authentication on page 44 for details.

5. To execute SQL statements, select **SQL Commander**>**New SQL Commander**. A SQL Commander tab opens.
6. Select values for the following fields:
   - **Database Connection**: Select connection alias you provided for the connection from the drop-down menu.
   - **Schema**: Select the schema you want to execute queries against from the drop-down menu.
7. In the SQL Commander tab, enter SQL commands you want to execute; then select **SQL Commander**>**Execute**. For example:

   To select all of the rows from the `BLOGS` table:

   ```
   SELECT * BLOGS
   ```

   To select the details of a specified company:

   ```
   SELECT * FROM COMPANY WHERE COMPANYID = <company_id>
   ```

   See "Supported SQL statements and extensions" for the supported syntax used to execute SQL statements.

   Note: If you are fetching large sets of data, you may want to limit the results using the Max Rows and Max Chars fields.

You have successfully accessed your data with DbVisualizer.

3

Configuring and connecting

This section provides information on how to connect to your data store using either the JDBC Driver Manager or DataDirect JDBC data sources, as well as information on how to implement and use functionality supported by the driver.

After the driver has been installed and defined on your classpath, you can connect from your application to your data in either of the following ways.

- Using the JDBC `DriverManager` by specifying the connection URL in the `DriverManager.getConnection()` method.
- Creating a JDBC data source that can be accessed through the Java Naming Directory Interface (JNDI).

For details, see the following topics:

Setting the classpath

The driver must be defined on your CLASSPATH before you can connect. The CLASSPATH is the search string your Java Virtual Machine (JVM) uses to locate JDBC drivers on your computer. If the driver is not defined on your CLASSPATH, you will receive a `class not found` exception when trying to load the driver. Set your system CLASSPATH to include the driver jar file as shown, where *`install_dir`* is the path to your product installation directory.

```
install_dir/lib/60/hubspot.jar
```

Windows Example

```
CLASSPATH=.;C:\Program Files\Progress\DataDirect\JDBC\lib\60\hubspot.jar
```

UNIX Example

```
CLASSPATH=.:/opt/Progress/DataDirect/JDBC/lib/60/hubspot.jar
```

Connecting using the JDBC Driver Manager

One way to connect to a service is through the JDBC DriverManager using the `DriverManager.getConnection()` method. As the following examples show, this method specifies a string containing a connection URL.

OAuth 2.0 access token authentication

```
Connection conn = DriverManager.getConnection
  ("jdbc:datadirect:hubspot:AuthenticationMethod=OAuth2;
    AccessToken=abcDEF123ghi-456Jklmno789-Pqrst01234;");
```

OAuth 2.0 refresh token authentication

```
Connection conn = DriverManager.getConnection
  ("jdbc:datadirect:hubspot:AuthenticationMethod=oauth2;
    ClientID=ab123c45-def6-7g89-gh1i-m2345no67891;
    ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd;
    RefreshToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;");
```

Note: See OAuth 2.0 authentication on page 42 for details.

API key authentication

```
Connection conn = DriverManager.getConnection
("jdbc:datadirect:hubspot:AuthenticationMethod=URLParameter;
  APIKey=1234ab5cd-1a2b-3a4b-678uvw-xyz91011;");
```

Note: See API key authentication on page 44 for details.

Passing the connection URL

After setting the CLASSPATH, the required connection information needs to be passed in the form of a connection URL. The following example includes the properties required for connecting with OAuth 2.0 refresh token grant authentication.

Connection URL Syntax

The connection URL takes the following form:

```
jdbc:datadirect:hubspot:AuthenticationMethod=OAuth2;
ClientID=client_id;ClientSecret=client_secret;
RefreshToken=refresh_token;Scope=scope;[property=value[;...]];
```

where:

client_id

specifies the client ID key for your application when authenticating with OAuth 2.0.

client_secret

specifies the client secret for your application when authenticating with OAuth 2.0.

Important: The client secret is a confidential value used to authenticate the application to the server. To prevent unauthorized access, this value must be securely maintained.

refresh_token

specifies the refresh token used to either request a new access token or renew an expired access token.

Important: The refresh token is a confidential value used to authenticate to the server. To prevent unauthorized access, this value must be securely maintained.

scope

(optional) specifies a space-separated list of OAuth scopes that limit the permissions granted by an access token.

property=value

specifies connection property settings. Multiple properties are separated by a semi-colon.

The following example connection string includes the properties required for connecting with the OAuth 2.0 refresh token grant.

```
Connection conn = DriverManager.getConnection
  ("jdbc:datadirect:hubspot:AuthenticationMethod=OAuth2;
    ClientID=ab123c45-def6-7g89-gh1i-m2345no67891;
    ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd;
    RefreshToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;");
```

See also

Connection property descriptions on page 45
Connection URL examples on page 12
OAuth 2.0 authentication on page 42

Generating connection URLs with the Configuration Manager

The driver includes a browser-based tool, Progress DataDirect HubSpot Configuration Manager, that allows you to generate connection URLs, test connections, and execute test queries. This section will guide you through generating and testing a connection URL that can be used by your application.

To generate a connection URL:

1. Open the HubSpot Configuration Manager by double-clicking the driver jar file. Or, in a command line, navigate to the directory containing your driver jar file; then, execute the following command:

   ```
   java -cp hubspot.jar
   ```

   The HubSpot Configuration Manager opens in your default web browser.

2. From the browser window, provide values of the connection properties you want to configure in the corresponding fields. A connection URL will generate in the Connection String field as you provide settings. To view more properties, select the tabs at the top of the page. See "Connection URL examples" for a list of required properties and properties used for different configurations.

   Note: If you do not specify a value for an optional property, the property will be omitted from the string and the default value will be used.

3. Optionally, you can manually edit your string by clicking the Edit button ().
4. At any point during the process, you can click **Test Connect** to attempt to connect to the service using the string generated in the Connection String field. In the **Test Connection** window:
   a) Provide values for any fields required by your service.
   b) Optionally, in the Test Query field, enter any SQL queries you want to execute during the test. For example:

      ```
      SELECT * FROM INFORMATION_SCHEMA.SYSTEM_TABLES
      ```

   c) Click **Execute**.

      If successful, the window displays a confirmation message and, if a query was specified, the results of the query.

   Note: The information you enter in the logon dialog box during a test connect is not saved.

5. To use your string, click the Copy button () and paste the string to a location that can be used by your application.
6. Optionally, click **Save** to store your connection string for later use.

See also

Connection URL examples on page 12

Testing connections and queries

You can quickly test a connection string and queries using Progress DataDirect HubSpot Configuration Manager.

To test your connection and query:

1. Open the HubSpot Configuration Manager by double-clicking on the driver jar file. Or, in a command line, navigate to the directory containing your driver jar file; then, execute the following command:

   ```
   java -cp hubspot.jar
   ```

   The HubSpot Configuration Manager opens in your default web browser.

2. Provide a connection string to test using one of the following methods:

   - Entering a string: Click the Edit button (); then, paste your string into the Connection String field. If you prefer, you can also type a string directly into this field.
   - Generating a string: Provide values of the connection properties you want to configure into the corresponding fields. The HubSpot Configuration Manager will generate a string in the Connection String field based on the values you specify.

3. Click **Test Connect** to attempt to connect to the service using the string specified in the Connection String field. The **Test Connection** window appears.
4. Provide connection property values for any fields required by your service.
5. To execute a test query with the test connection, enter a SQL query into the Test Query field. For example:

   ```
   SELECT * FROM INFORMATION_SCHEMA.SYSTEM_TABLES
   ```

6. Click **Execute**.

   If successful, the window displays a confirmation message and, if a query was specified, the results of the query.

Connecting using data sources

A *JDBC data source* is a Java object, specifically a `DataSource` object, that defines connection information required for a JDBC driver to connect to the database. Each JDBC driver vendor provides their own data source implementation for this purpose. A Progress DataDirect data source is Progress DataDirect's implementation of a `DataSource` object that provides the connection information needed for the driver to connect to a database.

Because data sources work with the Java Naming Directory Interface (JNDI) naming service, data sources can be created and managed separately from the applications that use them. Because the connection information is defined outside of the application, the effort to reconfigure your infrastructure when a change is made is minimized. For example, if the database is moved to another database server, the administrator need only change the relevant properties of the `DataSource` object. The applications using the database do not need to change because they only refer to the name of the data source.

How data sources are implemented

Data sources are implemented through a `DataSource` class. A data source class implements the following interfaces.

- `javax.sql.DataSource`
- `javax.sql.ConnectionPoolDataSource` (allows applications to use connection pooling)

Refer to "Connection Pool Manager" in the *Progress DataDirect for JDBC Drivers Reference* for more information.

See also

Driver and DataSource classes on page 12

Creating data sources

The following example files provide details on creating and using Progress DataDirect data sources with the Java Naming Directory Interface (JNDI), where *`install_dir`* is the product installation directory.

- *`install_dir`*`/Examples/JNDI/JNDI_LDAP_Example.java` can be used to create a JDBC data source and save it in your LDAP directory using the JNDI Provider for LDAP.
- *`install_dir`*`/Examples/JNDI/JNDI_FILESYSTEM_Example.java` can be used to create a JDBC data source and save it in your local file system using the File System JNDI Provider.

See "Example data source" for an example data source definition for the example files.

To connect using a JNDI data source, the driver needs to access a JNDI data store to persist the data source information. For a JNDI file system implementation, you must download the File System Service Provider from the Oracle Technology Network Java SE Support downloads page, unzip the files to an appropriate location, and add the `fscontext.jar` and `providerutil.jar` files to your CLASSPATH. These steps are not required for LDAP implementations because the LDAP Service Provider is included with supported versions of Java SE.

Example data source

To configure a data source using the example files, you will need to create a data source definition. The content required to create a data source definition is divided into three sections.

First, you will need to import the data source class. For example:

```
import com.ddtek.jdbcx.hubspot.HubSpotDataSource;
```

Next, you will need to set the values and define the data source. For example, the following definition contains the minimum properties required for a connection using the OAuth 2.0 refresh token authentication.

Note: Setting the client secret using a data source is generally not recommended. The data source persists all properties, including the ClientSecret property, in clear text.

```
HubSpotDataSource mds = new HubSpotDataSource();
mds.setDescription("My HubSpot Data Source");
mds.AuthenticationMethod("OAuth2");
mds.ClientID("ab123c45-def6-7g89-gh1i-m2345no67891");
mds.ClientSecret("12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd");
mds.RefreshToken("12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831");
```

Finally, you will need to configure the example application to print out the data source attributes. Note that this code is specific to the driver and should only be used in the example application. For example, you would add the following section for the minimum properties required to establish a connection:

```
if (ds instance of HubSpotDataSource)
{
HubSpotDataSource jmds = (HubSpotDataSource) ds;
System.out.println("description=" + jmds.getDescription());
System.out.println("authenticationmethod=" + jmds.getAuthenticationMethod());
...
System.out.println();
}
```

Calling a data source in an application

Applications can call a Progress DataDirect data source using a logical name to retrieve the `javax.sql.DataSource` object. This object loads the specified driver and can be used to establish a connection to the database.

Once the data source has been registered with JNDI, it can be used by your JDBC application as shown in the following code example.

```
Context ctx = new InitialContext();
DataSource ds = (DataSource)ctx.lookup("EmployeeDB");
Connection con = ds.getConnection("domino", "spark");
```

In this example, the JNDI environment is first initialized. Next, the initial naming context is used to find the logical name of the data source (`EmployeeDB`). The `Context.lookup()` method returns a reference to a Java object, which is narrowed to a `javax.sql.DataSource` object. Then, the `DataSource.getConnection()` method is called to establish a connection.

Testing a data source connection

You can use DataDirect Test™ to establish and test a data source connection. The screen shots in this section were taken on a Windows system.

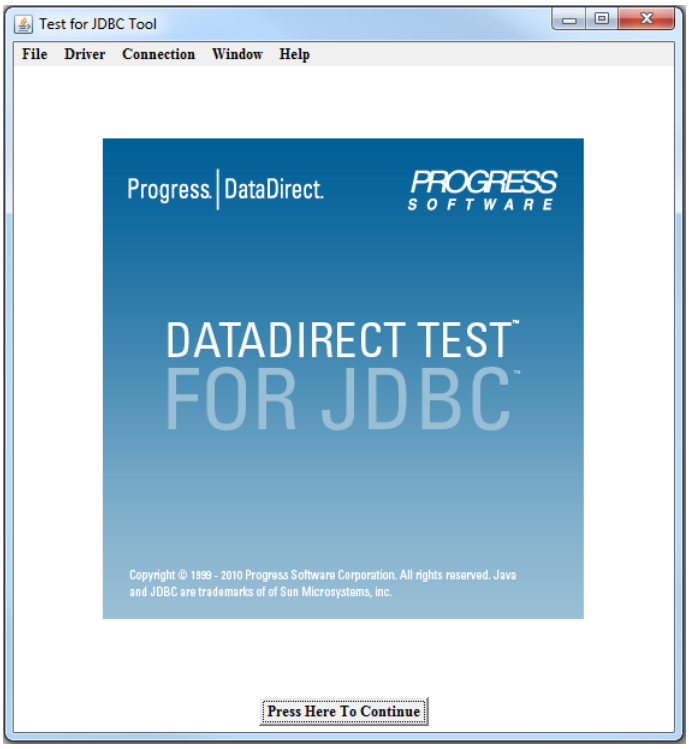
Take the following steps to establish a connection.

1. Navigate to the installation directory. The default location is:
   - Windows systems: `Program Files\Progress\DataDirect\JDBC\testforjdbc`
   - UNIX and Linux systems: `/opt/Progress/DataDirect/JDBC/testforjdbc`

   Note: For UNIX/Linux, if you do not have access to `/opt`, your home directory will be used in its place.

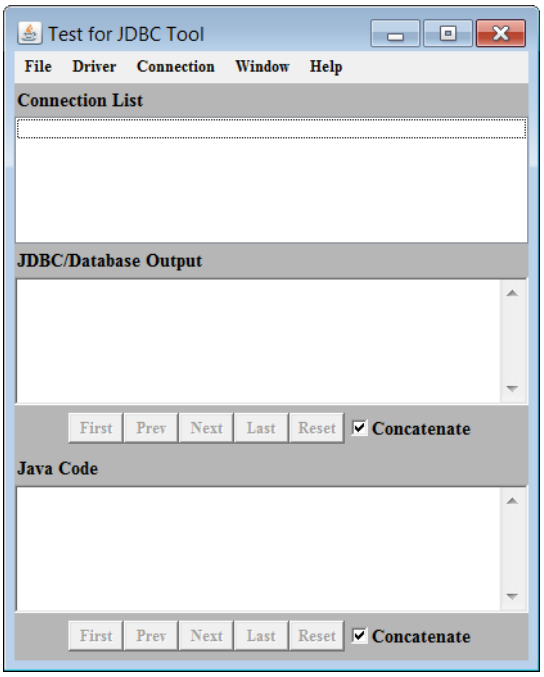
2. From the `testforjdbc` folder, run the platform-specific tool:
   - `testforjdbc.bat` (on Windows systems)
   - `testforjdbc.sh` (on UNIX and Linux systems)

   The **Test for JDBC Tool** window appears:



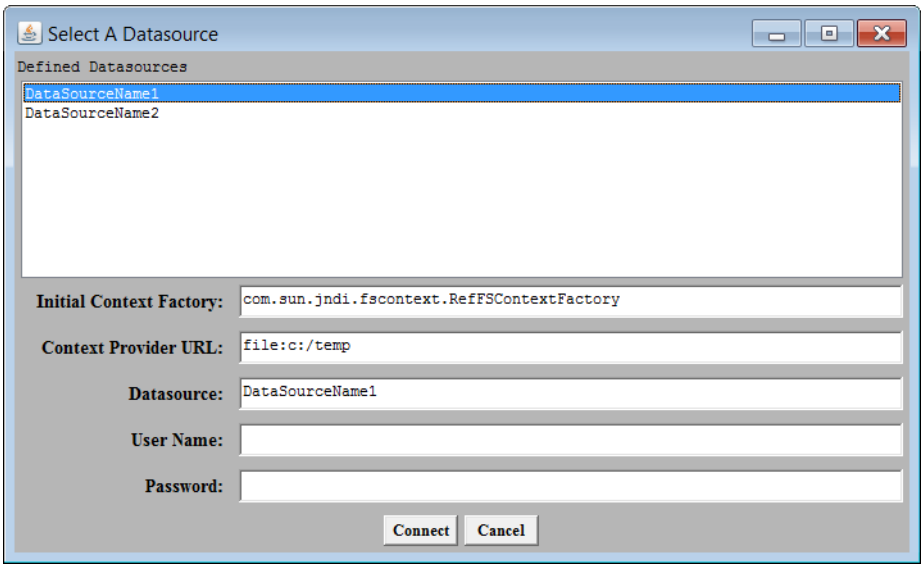
3. Click **Press Here to Continue**.

   The main dialog appears:



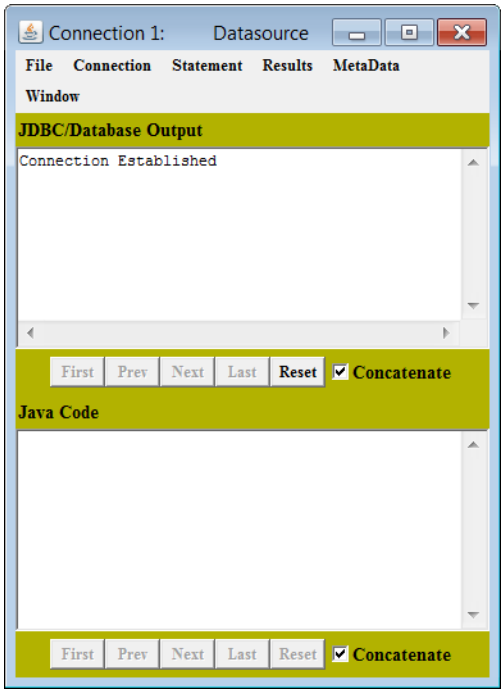
4. From the menu bar, select **Connection > Connect to DB via Data Source**.

   The **Select A Database** dialog appears:



5. Select a datasource template from the **Defined Datasources** field.
6. Provide the following information:
   a) In the **Initial Context Factory**, specify the location of the initial context provider for your application.
   b) In the **Context Provider URL**, specify the location of the context provider for your application.
   c) In the **Datasource** field, specify the name of your datasource.
7. If you are using user ID/password authentication, enter your user ID and password in the corresponding fields.
8. Click **Connect**.

If the connection information is entered correctly, the **JDBC/Database Output** window reports that a connection has been established. If a connection is not established, the window reports an error.



Performance considerations

FetchSize: FetchSize can be used to adjust the trade-off between throughput and response time. In general, setting larger values for FetchSize will improve throughput, but can reduce response time. You should set FetchSize to suit your environment. Smaller fetch sizes can improve the initial response time of the query. Larger fetch sizes improve overall fetch times at the cost of additional memory.

Authentication

The driver supports the following authentication methods:

- *OAuth 2.0 authentication* authenticates using OAuth 2.0 authentication flows.
- *API key authentication* authenticates using an API key.

By default, the driver is configured to use OAuth 2.0 authentication (`AuthenticationMethod=OAuth2`).

See also

AuthenticationMethod on page 49

OAuth 2.0 authentication

The driver supports the OAuth 2.0 grant types described in this section. The OAuth 2.0 properties you must specify depend on the grant type used in your environment. If you are unsure of the grant type or its requirements, contact your system administrator.

- Access token flow on page 42
- Refresh token grant on page 43

See also

Connection property descriptions on page 45

Access token flow

The access token authentication flow passes the access token directly from the client to the HubSpot service for authentication. Unlike other grant types, authentication credentials, such as authorization codes, are not exchanged in return for the access code. Instead, the access token is obtained from sources external to the flow and specified using the AccessToken property.

Note: Access tokens are temporary and must be replaced to maintain the session without interruption. The life of an access token is typically one hour.

To configure the driver to use an access token flow:

- Set the AuthenticationMethod property to `OAuth2`.

- Set the AccessToken property to the value of the access token obtained from external sources.

The following examples show the connection information required to establish a session using the access token flow.

Connection URL

```
Connection conn = DriverManager.getConnection
  ("jdbc:datadirect:hubspot:
    AuthenticationMethod=OAuth2;
    AccessToken=C3TQH9zjwek4CgJCU-4Mxb2DxLNfI2LB3a-dNfpWYx;");
```

Data Source

```
HubSpotDataSource mds = new HubSpotDataSource();
mds.setDescription("My HubSpot Data Source");
mds.setAuthenticationMethod("OAuth2");
mds.setAccessToken("C3TQH9zjweek4CgJCU-4Mxb2DxPLNfI2LB3a-dNfpWnYx");
```

See also

OAuth 2.0 access token flow on page 13
OAuth 2.0 properties on page 45

Refresh token grant

The refresh token grant is used to replace expired access tokens with active ones by exchanging the refresh token at the endpoint specified by the TokenURI property.

To configure the driver to use the refresh token grant:

- Set the AuthenticationMethod property to `OAuth2`.
- Set the ClientID property to specify the client ID key for your application.
- Set the ClientSecret property to specify the client secret for your application.

  Important: The client secret is a confidential value used to authenticate the application to the server. To prevent unauthorized access, this value must be securely maintained.

- Set the RefreshToken property to specify the refresh token used to request a new access token or renew an expired one.

  Important: The refresh token is a confidential value used to authenticate to the server. To prevent unauthorized access, this value must be securely maintained.

- Optionally, set the Scope property to specify a space-separated list of OAuth scopes to limit the permissions granted by the access token.

The following examples show the connection information required to establish a session using the refresh token grant.

Connection URL

```
Connection conn = DriverManager.getConnection
  ("jdbc:datadirect:hubspot:
    AuthenticationMethod=OAuth2;ClientID=ab123c45-def6-7g89-gh1i-m2345no67891;
    ClientSecret=12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd;
    RefreshToken=12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831;");
```

Data Source

```
HubSpotDataSource mds = new HubSpotDataSource();
mds.setDescription("My HubSpot Data Source");
mds.AuthenticationMethod("OAuth2");
mds.ClientID("ab123c45-def6-7g89-gh1i-m2345no67891");
mds.ClientSecret("12a3=bCD/EfGh4Ijk+Lm5P67qR8s=//TuV+WXy1Zabcd");
mds.RefreshToken("12a3=bCD/EfGh4Ijk+Lgd8g-44tk3c527831");
```

See also

OAuth 2.0 refresh token grant on page 14
OAuth 2.0 properties on page 45

API key authentication

The driver supports API key authentication. It allows you to authenticate to the REST endpoints using an API key. To learn how to generate an API key for your account, refer to the HubSpot documentation.

To configure the driver to use the API key authentication:

- Set the AuthenticationMethod property to `URLParameter`.
- Set the APIKey property to specify the API key.

The following examples show the connection information required to establish a session using the API key authentication.

Connection URL

```
Connection conn = DriverManager.getConnection
  ("jdbc:datadirect:hubspot:
    AuthenticationMethod=URLParameter;
    APIKey=12a3bCD-EfGh4Ijk-Lgd8g-44tk3c527831;");
```

Data Source

```
HubSpotDataSource mds = new HubSpotDataSource();
mds.setDescription("My HubSpot Data Source");
mds.AuthenticationMethod("URLParameter");
mds.APIKey("12a3bCD-EfGh4Ijk-Lgd8g-44tk3c527831");
```

See also

OAuth 2.0 refresh token grant on page 14
OAuth 2.0 properties on page 45

4

Connection property descriptions

You can use connection properties to customize the driver for your environment. This section organizes connection properties according to functionality. You can use connection properties with either the JDBC `DriverManager` or a JDBC data source. For a `DriverManager` connection, a property is expressed as a key value pair and takes the form *`property=value`*. For a data source connection, a property is expressed as a JDBC method and takes the form `set`*`property`*`(`*`value`*`)`.

Note: Connection property names are case-insensitive. For example, `Password` is the same as `password`.

Note: `setUser("abc@defcorp.com")`.

The following tables describe the connection properties by functionality.In a JDBC data source, string values must be enclosed in double quotation marks, for example,

- OAuth 2.0 properties
- API key properties
- Proxy server properties
- Web service and timeout properties
- In a JDBC data source, string values must be enclosed inAdditional properties

OAuth 2.0 properties

The following table summarizes properties used for OAuth 2.0 authentication. The OAuth 2.0 properties you must specify depend on the grant type used in your environment. If you are unsure of the grant type or its requirements, contact your system administrator. For details on supported grant types, see OAuth 2.0 authentication on page 42.

Property	Data Source Method	Default
AccessToken on page 48	`getAccessToken()` `setAccessToken(String)`	No default value
AuthenticationMethod on page 49	`getAuthenticationMethod()` `setAuthenticationMethod(String)`	No default value
ClientID on page 50	`getClientId()` `setClientId(String)`	No default value
ClientSecret on page 51	`getClientSecret()` `setClientSecret(String)`	No default value
RefreshToken on page 57	`getRefreshToken()` `setRefreshToken(String)`	No default value
Scope on page 58	`getScope()` `setScope(String)`	No default value
TokenURI on page 58	`getTokenUri()` `setTokenUri(String)`	No default value

API key properties

The following table summarizes the connection properties required for API key authentication.

Property	Data Source Method	Default
AuthenticationMethod on page 49	`getAuthenticationMethod()` `setAuthenticationMethod(String)`	`OAuth2`
APIKey on page 49	`getApiKey()` `setApiKey(String)`	No default value

Proxy server properties

The following table summarizes proxy server connection properties.

Property	Data Source Method	Default
ProxyHost on page 54	`getProxyHost()` `setProxyHost(String)`	No default value

Property	Data Source Method	Default
ProxyPassword on page 55	`getProxyPassword()` `setProxyPassword(String)`	No default value
ProxyPort on page 56	`getProxyPort()` `setProxyPort(Integer)`	No default value
ProxyUser on page 56	`getProxyUser()` `setProxyUser(String)`	No default value

Web service and timeout properties

The following table summarizes web service connection properties, including those related to timeouts.

Property	Data Source Method	Default
WSRetryCount on page 59	`getWSRetryCount()` `setWSRetryCount(Integer)`	`5`
WSTimeout on page 60	`getWSTimeout()` `setWsTimeout(Integer)`	`120` (seconds)

Additional properties

The following table summarizes additional connection properties.

Property	Data Source Method	Default
FetchSize on page 53	`getFetchSize()` `setFetchSize(Integer)`	`100` (rows)
LogConfigFile on page 54	`getLogConfigFile()` `setLogConfigFile(String)`	`ddlogging.properties`

For details, see the following topics:

- AccessToken
- APIKey
- AuthenticationMethod
- ClientID
- ClientSecret
- DebugRecord

AccessToken

Purpose

Specifies the access token used to authenticate to HubSpot with OAuth 2.0 enabled. Typically, this property is configured by the application; however, in some scenarios, you may need to secure a token using external processes. In those instances, you can also use this property to set the access token manually.

Valid Values

String

where:

String

is an access token you have obtained from the authentication service.

Notes

- Access tokens expire ten minutes after generation. Once connected, the access token remains valid till the session is disconnected.
- See "OAuth 2.0 authentication" for examples and more information.

Data Source Methods

```
public String getAccessToken()
```

```
public void setAccessToken(String)
```

Default Value

No default value

Data Type

String

See also

OAuth 2.0 authentication on page 42
Access token flow on page 42

APIKey

Purpose

Specifies the API key used to authenticate to the REST endpoints when API key authentication is enabled (`AuthenticationMethod=URLParameter`). To learn how to generate an API key for your account, refer to the HubSpot documentation.

Valid Values

`String`

where:

`String`

is the API key you have generated for your account.

Data Source Methods

`public String getApiKey()`

`public void setApiKey(String)`

Default Value

No default value

Data Type

String

See also

API key authentication on page 44

AuthenticationMethod

Purpose

Determines which authentication mechanism the driver uses when establishing a connection.

Valid Values

`OAuth2` | `URLParameter`

Behavior

If set to `OAuth2`, the driver uses OAuth 2.0 to authenticate to the REST endpoints. See "OAuth 2.0 authentication" for details.

If set to `URLParameter`, the driver uses an API key to authenticate to the REST endpoints. See "API key authentication" for details.

Data Source Methods

```
public String getAuthenticationMethod()
```

```
public void setAuthenticationMethod(String)
```

Default Value

`OAuth2`

Data Type

String

See also

OAuth 2.0 authentication on page 42
API key authentication on page 44

ClientID

Purpose

Specifies the client ID key for your application when authenticating to HubSpot with OAuth 2.0 enabled.

Valid Values

`String`

where:

`String`

is the client ID key for your application.

Notes

See "OAuth 2.0 authentication" for more information.

Data Source Methods

```
public String getClientId()
```

```
public void setClientId(String)
```

Default Value

No default value

Data Type

String

See also

OAuth 2.0 authentication on page 42

ClientSecret

Purpose

Specifies the client secret for your application when authenticating to HubSpot with OAuth 2.0 enabled.

Important: The client secret is a confidential value used to authenticate the application to the service. To prevent unauthorized access, this value must be securely maintained.

Valid Values

String

where:

String

is the client secret for your application.

Notes

See "OAuth 2.0 authentication" for more information.

Data Source Methods

```
public String getClientSecret()
```

```
public void setClientSecret(String)
```

Default Value

No default value

Data Type

String

See also

OAuth 2.0 authentication on page 42

DebugRecord

Purpose

Specifies the directory where the driver generates debug record files. When a value is specified, the driver records server requests and responses to a set of files stored in this location. These files assist in troubleshooting by providing a method for Technical Support to reproduce and debug issues for REST services that are not publicly accessible.

Important: Debug record files may capture security-related headers, such as auth or token headers. Before sending Technical Support debug files, review the content to remove any confidential information that may have been recorded.

Valid Values

debug_record_folder

where:

debug_record_folder

is the location of the folder where the debug record files are to be generated. For example, `C:\Temp\MyDebug Folder`.

Notes

- The specified directory must exist.
- You must have write access to the specified directory.
- The contents of the specified directory are deleted every time a connection is established.
- For more information, refer to "Enabling Debug Record Mode" in the *Progress DataDirect for JDBC Drivers Reference*.
- For assistance, contact Technical Support.

Data Source Methods

```
public String getDebugRecord()
```

```
public void setDebugRecord(String)
```

Default Value

No default value

Data Type

String

See also

Contacting Technical Support on page 27

FetchSize

Purpose

Specifies the maximum number of rows that the driver processes before returning data to the application when executing a Select. This value provides a suggestion to the driver as to the number of rows it should internally process before returning control to the application. The driver may fetch fewer rows to conserve memory when processing exceptionally wide rows.

Valid Values

`0` | *x*

where:

x

is a positive integer indicating the number of rows that should be processed.

Behavior

If set to `0`, the driver processes all the rows of the result before returning control to the application. When large data sets are being processed, setting FetchSize to `0` can diminish performance and increase the likelihood of out-of-memory errors.

If set to *x*, the driver limits the number of rows that may be processed for each fetch request before returning control to the application.

Notes

- To optimize throughput and conserve memory, the driver uses an internal algorithm to determine how many rows should be processed based on the width of rows in the result set. Therefore, the driver may process fewer rows than specified by FetchSize when the result set contains exceptionally wide rows. Alternatively, the driver processes the number of rows specified by FetchSize when the result set contains rows of unexceptional width.
- FetchSize can be used to adjust the trade-off between throughput and response time. Smaller fetch sizes can improve the initial response time of the query. Larger fetch sizes can improve overall response times at the cost of additional memory.
- You can use FetchSize to reduce demands on memory and decrease the likelihood of out-of-memory errors. Simply, decrease FetchSize to reduce the number of rows the driver is required to process before returning data to the application.

Data Source Methods

```
public Integer getFetchSize()
```

```
public void setFetchSize(Integer)
```

Default Value

`100`

Data Type

Integer

See also

Performance considerations on page 42

LogConfigFile

Purpose

Specifies the file name, and optionally, the path of the properties file used to initialize driver logging.

Valid Values

String

where:

String

is the relative or fully qualified path of the properties file to load to initialize driver logging. If you do not specify a path, the driver looks for this file in the current working directory. If the specified file does not exist, the driver continues searching for an appropriate properties file as described in "Using Java Logging" in the *Progress DataDirect for JDBC Drivers Reference*.

Data Source Methods

```
public String getLogConfigFile()
```

```
public void setLogConfigFile(String)
```

Default Value

```
ddlogging.properties
```

Data Type

String

ProxyHost

Purpose

Identifies a proxy server to use for the first connection.

Valid Values

server_name | *IP_address*

where:

server_name

is the name of the proxy server, which may be qualified with the domain name.

IP_address

is an IP address, specified in either IPv4 or IPv6 format, or a combination of the two.

Data Source Methods

`public String getProxyHost()`

`public void setProxyHost(String)`

Default Value

No default value

Data Type

String

See also

Proxy server on page 15

ProxyPassword

Purpose

Specifies the password needed to connect to a proxy server for the first connection.

Valid Values

`password`

where:

`password`

is a valid password for that server. Contact your system administrator to obtain a valid password.

Data Source Methods

`public String getProxyPassword()`

`public void setProxyPassword(String)`

Default Value

No default value

Data Type

String

See also

Proxy server on page 15

ProxyPort

Purpose

Specifies the port number where the proxy server is listening for HTTP or HTTPS requests for the first connection.

Valid Values

port

where:

port

is the port number on which the proxy server is listening. Contact your system administrator to obtain the correct port.

Data Source Methods

```
public Integer getProxyPort()
```

```
public void setProxyPort(Integer)
```

Default Value

0 which means that the default value is determined by whether the value specified for the ProxyHost property is an HTTP or HTTPS URL.

For HTTP: 80

For HTTPS: 443

Data Type

Integer

See also

Proxy server on page 15

ProxyUser

Purpose

Specifies the user name needed to connect to a proxy server for the first connection.

Valid Values

user_name

where:

user_name

is a valid user ID for the proxy server.

Data Source Methods

```
public String getProxyUser()
```

```
public void setProxyUser(String)
```

Default Value

No default value

Data Type

String

See also

Proxy server on page 15

RefreshToken

Purpose

Specifies the refresh token used to either request a new access token or renew an expired access token for OAuth 2.0 implementations.

Important: The refresh token is a confidential value used to authenticate to the service. To prevent unauthorized access, this value must be securely maintained.

Valid Values

String

where:

String

is the refresh token you have obtained from the HubSpot service.

Notes

- See "OAuth 2.0 authentication" for more information.

Data Source Methods

```
public String getRefreshToken()
```

```
public void setRefreshToken(String)
```

Default Value

No default value

Data Type

String

See also

OAuth 2.0 authentication on page 42

Scope

Purpose

Specifies a space-separated list of OAuth scopes that limit the permissions granted by an access token.

Valid Values

String

where:

String

is a space-separated list of security scopes.

Data Source Methods

`public String getScope()`

`public void setScope(String)`

Default Value

No default value

Data Type

String

See also

OAuth 2.0 refresh token grant on page 14
Refresh token grant on page 43

TokenURI

Purpose

Specifies the endpoint used to exchange authentication credentials for access tokens when OAuth 2.0 authentication is enabled.

Valid Values

String

where:

String

is the endpoint used to retrieve OAuth 2.0 access tokens.

Notes

See "OAuth 2.0 authentication" for more information.

Data Source Methods

`public String getTokenUri()`

`public void setTokenUri(String)`

Default Value

`https://api.hubapi.com/oauth/v1/token`

Data Type

String

See also

Refresh token grant on page 43

WSRetryCount

Purpose

Specifies the number of times the driver retries a timed-out Select request. The timeout period is specified by the WSTimeout connection property.

Valid Values

`0` | *x*

where:

x

is a positive integer

Behavior

If set to `0`, the driver does not retry timed-out requests after the initial unsuccessful attempt.

If set to *x*, the driver retries the timed-out request the specified number of times.

Data Source Methods

`public Integer getWSRetryCount()`

`public void setWSRetryCount(Integer)`

Default Value

5

Data Type

Integer

See also

WSTimeout on page 60

WSTimeout

Purpose

Specifies the time, in seconds, that the driver waits for a response to a web service request.

Valid Values

`0` | *x*

where:

x

is a positive integer that defines the number of seconds the driver waits for a response to a web service request.

Behavior

If set to `0`, the driver waits indefinitely for a response; there is no timeout.

If set to *x*, the driver uses the value as the default timeout, measured in seconds, for any statement created by the connection.

If a Select request times out and WSRetryCount is set to retry timed-out requests, the driver retries the request the specified number of times.

Data Source Methods

```
public Integer getWSTimeout()
```

```
public void setWSTimeout(Integer)
```

Default Value

`120`

Data Type

Integer

See also

WSRetryCount on page 59

5

Supported SQL statements and extensions

The driver provides support for the SQL statements and the SQL extensions described in this section. SQL extensions are denoted by an (EXT) in the topic title.

For details, see the following topics:

Alter Session (EXT)

Purpose

Changes various attributes of a local or remote session. A local session maintains the state of the overall connection. A remote session maintains the state that pertains to a particular remote data source connection.

Syntax

`ALTER SESSION SET` *`attribute_name=value`*

where:

attribute_name

specifies the name of the attribute to be changed. Attributes apply to either local or remote sessions.

value

specifies the value for that attribute.

The following table lists the local and remote session attributes, and provides descriptions of each.

Table 4: Alter Session Attributes

Attribute Name	Session Type	Description
Current_Schema	Local	Sets the current schema for the local session. The current schema is the schema used when an identifier in a SQL statement is unqualified. The string value must be the name of a schema visible in the local session. For example: `ALTER SESSION SET CURRENT_SCHEMA=`
Stmt_Call_Limit	Local	Sets the maximum number of Web service calls the driver can make in executing a statement. Setting the Stmt_Call_Limit attribute has the same effect as setting the Statement Call Limit connection option. It sets the default Web service call limit used by any statement on the connection. Executing this command on a statement overrides the previously set Statement Call Limit for the connection. The value specified must be a positive integer or `0`. The value `0` means that no call limit exists. For example: `ALTER SESSION SET STMT_CALL_LIMIT=150`

Ws_Call_Count	Remote	Resets the Web service call count of a remote session to the value specified. The value must be `0` or a positive integer. WS_Call_Count represents the total number of Web service calls made to the remote data source instance for the current session. For example:
		`ALTER SESSION SET .WS_CALL_COUNT=0`
		The current value of WS_Call_Count can be obtained by referring to the System_Remote_Sessions system table (see SYSTEM_REMOTE_SESSIONS Catalog Table for details). For example:
		`SELECT * from information_schema.system_remote_sessions WHERE session_id = cursessionid()`

Select

Purpose

Use the Select statement to fetch results from one or more tables.

Syntax

```
SELECT select_clause from_clause
[where_clause]
[groupby_clause]
[having_clause]
[{UNION [ALL | DISTINCT] |
 {MINUS [DISTINCT] | EXCEPT [DISTINCT]} |
 INTERSECT [DISTINCT]} select_statement]
[limit_clause]
```

where:

select_clause

specifies the columns from which results are to be returned by the query. See "Select clause" for a complete explanation.

from_clause

specifies one or more tables on which the other clauses in the query operate. See "From clause" for a complete explanation.

where_clause

is optional and restricts the results that are returned by the query. See "Where clause" for a complete explanation.

groupby_clause

is optional and allows query results to be aggregated in terms of groups. See "Group By clause" for a complete explanation.

having_clause

is optional and specifies conditions for groups of rows (for example, display only the departments that have salaries totaling more than $200,000). See "Having clause" for a complete explanation.

UNION

is an optional operator that combines the results of the left and right Select statements into a single result. See "Union operator" for a complete explanation.

INTERSECT

is an optional operator that returns a single result by keeping any distinct values from the results of the left and right Select statements. See "Intersect operator" for a complete explanation.

EXCEPT | MINUS

are synonymous optional operators that returns a single result by taking the results of the left Select statement and removing the results of the right Select statement. See "Except and Minus operators" for a complete explanation.

orderby_clause

is optional and sorts the results that are returned by the query. See "Order By clause" for a complete explanation.

limit_clause

is optional and places an upper bound on the number of rows returned in the result. See "Limit clause" for a complete explanation.

Select clause

Purpose

Use the Select clause to specify with a list of column expressions that identify columns of values that you want to retrieve or an asterisk (*) to retrieve the value of all columns.

Syntax

```
SELECT [{LIMIT offsetnumber | TOP number}] [ALL | DISTINCT] {* | column_expression
[[AS] column_alias] [,column_expression [[AS] column_alias], ...]}
```

where:

LIMIT *offset number*

creates the result set for the Select statement first and then discards the first number of rows specified by *offset* and returns the number of remaining rows specified by *number*. To not discard any of the rows, specify 0 for *offset*, for example, LIMIT 0 *number*. To discard the first *offset* number of rows and return all the remaining rows, specify 0 for *number*, for example, LIMIT *offset*0.

TOP *number*

is equivalent to LIMIT 0*number*.

column_expression

can be simply a column name (for example, last_name). More complex expressions may include mathematical operations or string manipulation (for example, salary * 1.05). See "SQL expressions" for details. *column_expression* can also include aggregate functions. See "Aggregate functions" for details.

column_alias

can be used to give the column a descriptive name. For example, to assign the alias department to the column dep:

```
SELECT dep AS department FROM emp
```

DISTINCT

eliminates duplicate rows from the result of a query. This operator can precede the first column expression. For example:

```
SELECT DISTINCT dep FROM emp
```

Notes

- Separate multiple column expressions with commas (for example, SELECT last_name, first_name, hire_date).

- Column names can be prefixed with the table name or table alias. For example, `SELECT emp.last_name or e.last_name`, where `e` is the alias for the table `emp`.
- NULL values are not treated as distinct from each other. The default behavior is that all result rows be returned, which can be made explicit with the keyword `ALL`.

See also

SQL expressions on page 73

Aggregate functions

Aggregate functions can also be a part of a Select clause. Aggregate functions return a single value from a set of rows. An aggregate can be used with a column name (for example, `AVG(salary)`) or in combination with a more complex column expression (for example, `AVG(salary * 1.07)`).

The following table lists supported aggregate functions.

Note: Doubly nested aggregates, such as `SUM(COUNT(col1))`, are currently not permitted by the driver.

Table 5: Aggregate Functions

Aggregate	Returns
AVG	The average of the values in a numeric column expression. For example, `AVG(salary)` returns the average of all salary column values.
COUNT	The number of values in any field expression. For example, `COUNT(name)` returns the number of name values. When using `COUNT` with a field name, `COUNT` returns the number of non-NULL column values. A special example is `COUNT(*)`, which returns the number of rows in the set, including rows with NULL values. **Note:** The driver does not support `COUNT(DISTINCT *)`. For example, `SELECT COUNT(DISTINCT *) FROM mytable` results in a syntax error.
MAX	The maximum value in any column expression. For example, `MAX(salary)` returns the maximum salary column value.
MIN	The minimum value in any column expression. For example, `MIN(salary)` returns the minimum salary column value.
SUM	The total of the values in a numeric column expression. For example, `SUM(salary)` returns the sum of all salary column values.

Example

The following example uses the `COUNT`, `MAX`, and `AVG` aggregate functions:

```
SELECT
     COUNT(amount) AS numOpportunities,
     MAX(amount) AS maxAmount,
     AVG(amount) AS avgAmount
FROM opportunity o INNER JOIN user u
     ON o.ownerId = u.id
```

```
WHERE o.isClosed = 'false' AND
     u.name = 'MyName'
```

From clause

Purpose

The From clause indicates the tables to be used in the Select statement.

Syntax

`FROM` *`table_name`* `[`*`table_alias`*`] [,...]`

where:

`table_name`

is the name of a table or a subquery. Multiple tables define an implicit inner join among those tables. Multiple table names must be separated by a comma. For example:

```
SELECT * FROM emp, dep
```

Subqueries can be used instead of table names. Subqueries must be enclosed in parentheses. See "Subquery in a From clause" for an example.

`table_alias`

is a name used to refer to a table in the rest of the Select statement. When you specify an alias for a table, you can prefix all column names of that table with the table alias.

Example

The following example specifies two table aliases, `e` for `emp` and `d` for `dep`:

```
SELECT e.name, d.deptName
FROM emp e, dep d
WHERE e.deptId = d.id
```

`table_alias` is a name used to refer to a table in the rest of the Select statement. When you specify an alias for a table, you can prefix all column names of that table with the table alias. For example, given the table specification:

```
FROM emp E
```

you may refer to the last_name field as E.last_name. Table aliases must be used if the Select statement joins a table to itself. For example:

```
SELECT * FROM emp E, emp F WHERE E.mgr_id = F.emp_id
```

The equal sign (=) includes only matching rows in the results.

Join in a From clause

Purpose

You can use a Join as a way to associate multiple tables within a Select statement. Joins may be either explicit or implicit. For example, the following is the example from the previous section restated as an explicit inner join:

```
SELECT * FROM emp INNER JOIN dep ON id=empId
SELECT e.name, d.deptName
FROM emp e INNER JOIN dep d ON e.deptId = d.id;
```

whereas the following is the same statement as an implicit inner join:

```
SELECT * FROM emp, dep WHERE emp.deptID=dep.id
```

Note: The `ON` clause in a join expression must evaluate to a true or false value.

Syntax

```
FROM table_name {RIGHT OUTER | INNER | LEFT OUTER | CROSS | FULL OUTER} JOIN table.key
 ON search-condition
```

Example

In this example, two tables are joined using `LEFT OUTER JOIN`. `T1`, the first table named includes nonmatching rows.

```
SELECT * FROM T1 LEFT OUTER JOIN T2 ON T1.key = T2.key
```

If you use a `CROSS JOIN`, no `ON` expression is allowed for the join.

Subquery in a From clause

Subqueries can be used in the From clause in place of table references (*`table_name`*).

Example

```
SELECT * FROM (SELECT * FROM emp WHERE sal > 10000) new_emp, dept WHERE
new_emp.deptno = dept.deptno
```

See also

Subqueries on page 81

Where clause

Purpose

Specifies the conditions that rows must meet to be retrieved.

Syntax

`WHERE` *`expr1 rel_operator expr2`*

where:

expr1

is either a column name, literal, or expression.

expr2

is either a column name, literal, expression, or subquery. Subqueries must be enclosed in parentheses.

rel_operator

is the relational operator that links the two expressions.

Example

The following Select statement retrieves the first and last names of employees that make at least $20,000.

```
SELECT last_name, first_name FROM emp WHERE salary >= 20000
```

See also

SQL expressions on page 73
Subqueries on page 81

Group By clause

Purpose

Specifies the names of one or more columns by which the returned values are grouped. This clause is used to return a set of aggregate values.

Syntax

```
GROUP BY column_expression [,...]
```

where:

column_expression

is either a column name or a SQL expression. Multiple values must be separated by a comma. If *column_expression* is a column name, it must match one of the column names specified in the Select clause. Also, the Group By clause must include all non-aggregate columns specified in the Select list.

Example

The following example totals the salaries in each department:

```
SELECT dept_id, sum(salary) FROM emp GROUP BY dept_id
```

This statement returns one row for each distinct department ID. Each row contains the department ID and the sum of the salaries of the employees in the department.

See also

SQL expressions on page 73
Subqueries on page 81

Having clause

Purpose

Specifies conditions for groups of rows (for example, display only the departments that have salaries totaling more than $200,000). This clause is valid only if you have already defined a Group By clause.

Syntax

`HAVING` *`expr1 rel_operator expr2`*

where:

`expr1 | expr2`

can be column names, constant values, or expressions. These expressions do not have to match a column expression in the Select clause. See "SQL expressions" for details regarding SQL expressions.

`rel_operator`

is the relational operator that links the two expressions.

Example

The following example returns only the departments that have salaries totaling more than $200,000:

```
SELECT dept_id, sum(salary) FROM emp GROUP BY dept_id HAVING sum(salary) > 200000
```

See also

SQL expressions on page 73
Subqueries on page 81

Union operator

Purpose

Combines the results of two Select statements into a single result. The single result is all the returned rows from both Select statements. By default, duplicate rows are not returned. To return duplicate rows, use the All keyword (`UNION ALL`).

Syntax

```
select_statement
UNION [ALL | DISTINCT] | {MINUS [DISTINCT] | EXCEPT [DISTINCT]} | INTERSECT
[DISTINCT]select_statement
```

Notes

- When using the Union operator, the Select lists for each Select statement must have the same number of column expressions with the same data types and must be specified in the same order.

Example A

The following example has the same number of column expressions, and each column expression, in order, has the same data type.

```
SELECT last_name, salary, hire_date FROM emp
UNION
SELECT name, pay, birth_date FROM person
```

Example B

The following example is *not* valid because the data types of the column expressions are different (`salary FROM emp` has a different data type than `last_name FROM raises`). This example does have the same number of column expressions in each Select statement but the expressions are not in the same order by data type.

```
SELECT last_name, salary FROM emp
UNION
SELECT salary, last_name FROM raises
```

Intersect operator

Purpose

Intersect operator returns a single result set. The result set contains rows that are returned by both Select statements. Duplicates are returned unless the Distinct operator is added.

Syntax

```
select_statement
INTERSECT [DISTINCT]
select_statement
```

where:

`DISTINCT`

eliminates duplicate rows from the results.

Notes

- When using the Intersect operator, the Select lists for each Select statement must have the same number of column expressions with the same data types and must be specified in the same order.

Example A

The following example has the same number of column expressions, and each column expression, in order, has the same data type.

```
SELECT last_name, salary, hire_date FROM emp
INTERSECT [DISTINCT]
SELECT name, pay, birth_date FROM person
```

Example B

The following example is *not* valid because the data types of the column expressions are different (`salary FROM emp` has a different data type than `last_name FROM raises`). This example does have the same number of column expressions in each Select statement but the expressions are not in the same order by data type.

```
SELECT last_name, salary FROM emp
INTERSECT
SELECT salary, last_name FROM raises
```

Except and Minus operators

Purpose

Return the rows from the left Select statement that are not included in the result of the right Select statement.

Syntax

```
select_statement
{EXCEPT [DISTINCT] | MINUS [DISTINCT]}
select_statement
```

where:

`DISTINCT`

eliminates duplicate rows from the results.

Notes

- When using one of these operators, the Select lists for each Select statement must have the same number of column expressions with the same data types and must be specified in the same order.

Example A

The following example has the same number of column expressions, and each column expression, in order, has the same data type.

```
SELECT last_name, salary, hire_date FROM emp
EXCEPT
SELECT name, pay, birth_date FROM person
```

Example B

The following example is *not* valid because the data types of the column expressions are different (`salary FROM emp` has a different data type than `last_name FROM raises`). This example does have the same number of column expressions in each Select statement but the expressions are not in the same order by data type.

```
SELECT last_name, salary FROM emp
EXCEPT
SELECT salary, last_name FROM raises
```

Order By clause

Purpose

The Order By clause specifies how the rows are to be sorted.

Syntax

```
ORDER BY sort_expression [DESC | ASC] [,...]
```

where:

sort_expression

is either the name of a column, a column alias, a SQL expression, or the positioned number of the column or expression in the select list to use.

The default is to perform an ascending (ASC) sort.

Example

To sort by `last_name` and then by `first_name`, you could use either of the following Select statements:

```
SELECT emp_id, last_name, first_name FROM emp
ORDER BY last_name, first_name
```

or

```
SELECT emp_id, last_name, first_name FROM emp
ORDER BY 2,3
```

In the second example, `last_name` is the second item in the Select list, so `ORDER BY 2,3` sorts by `last_name` and then by `first_name`.

See also

SQL expressions on page 73

Limit clause

Purpose

Places an upper bound on the number of rows returned in the result.

Syntax

```
LIMIT number_of_rows [OFFSET offset_number]
```

where:

number_of_rows

specifies a maximum number of rows in the result. A negative number indicates no upper bound.

`OFFSET`

specifies how many rows to skip at the beginning of the result set. *`offset_number`* is the number of rows to skip.

Notes

- In a compound query, the Limit clause can appear only on the final Select statement. The limit is applied to the entire query, not to the individual Select statement to which it is attached.

Example

The following example returns a maximum of 20 rows.

```
SELECT last_name, first_name FROM emp WHERE salary > 20000 ORDER BY dept_idc LIMIT
20
```

SQL expressions

An expression is a combination of one or more values, operators, and SQL functions that evaluate to a value. You can use expressions in the Where, and Having of Select statements; and in the Set clauses of Update statements.

Expressions enable you to use mathematical operations as well as character string manipulation operators to form complex queries.

The driver supports both unquoted and quoted identifiers. An unquoted identifier must start with an ASCII alpha character and can be followed by zero

Quoted identifiers must be enclosed in double quotation marks (""). A quoted identifier can contain any Unicode character including the space character. The driver recognizes the Unicode escape sequence \uxxxx as a Unicode character. You can specify a double quotation mark in a quoted identifier by escaping it with a double quotation mark.

The maximum length of both quoted and unquoted identifiers is 128 characters.

Valid expression elements are:

- Column names
- Literals
- Operators
- Functions

Column names

The most common expression is a simple column name. You can combine a column name with other expression elements.

Literals

Literals are fixed data values. For example, in the expression `PRICE * 1.05`, the value 1.05 is a constant. Literals are classified into types, including the following:

- Binary
- Character string
- Date
- Floating point
- Integer
- Numeric
- Time
- Timestamp

The following table describes the literal format for supported SQL data types.

Table 6: Literal Syntax Examples

SQL Type	Literal Syntax	Example
BIGINT	*n* where *n* is any valid integer value in the range of the INTEGER data type	`12 or -34 or 0`
BOOLEAN	Min Value: 0 Max Value: 1	`0` `1`
DATE	DATE'*date*'	`'2010-05-21'`
DATETIME	TIMESTAMP'*ts*'	`'2010-05-21 18:33:05.025'`

DECIMAL	*n.f*
	where:
	n
	is the integral part
	f
	is the fractional part
		`0.25`
		`3.1415`
		`-7.48`

DOUBLE	*n.f*E*x* where: *n* is the integral part *f* is the fractional part *x* is the exponent	`1.2E0` or `2.5E40` or `-3.45E2` or `5.67E-4`
INTEGER	*n* where *n* is a valid integer value in the range of the INTEGER data type	`12` or `-34` or `0`

SQL Type	Literal Syntax	Example
LONGVARBINARY	'*hex_value*'	'000482ff'
LONGVARCHAR	'*value*'	'This is a string literal'
TIME	TIME'*time*'	'2010-05-21 18:33:05.025'
VARCHAR	'*value*'	'This is a string literal'

Character string literals

Text specifies a character string literal. A character string literal must be enclosed in single quotation marks. To represent one single quotation mark within a literal, you must enter two single quotation marks. When the data in the fields is returned to the client, trailing blanks are stripped.

A character string literal can have a maximum length of 32 KB, that is, (32*1024) bytes.

Example

```
'Hello'
'Jim''s friend is Joe'
```

Numeric literals

Unquoted numeric values are treated as numeric literals. If the unquoted numeric value contains a decimal point or exponent, it is treated as a real literal; otherwise, it is treated as an integer literal.

Example

```
+1894.1204
```

Binary literals

Binary literals are represented with single quotation marks. The valid characters in a binary literal are 0-9, a-f, and A-F.

Example

```
'00af123d'
```

Date/Time literals

Date and time literal values are enclosed in single quotion marks ('*value*').

- The format for a Date literal is DATE'*date*'.
- The format for a Time literal is TIME'*time*'.
- The format for a Timestamp literal is TIMESTAMP'*ts*'.

Integer literals

Integer literals are represented by a string of numbers that are not enclosed in quotation marks and do not contain decimal points.

Notes

- Integer constants must be whole numbers; they cannot contain decimals.
- Integer literals can start with sign characters (+/-).

Example

```
1994 or -2
```

Operators

This section describes the operators that can be used in SQL expressions.

Note: Numeric operators are restricted to numeric types. Numeric operators do not support non-numeric types.

Unary operator

A unary operator operates on only one operand.

```
operator operand
```

Binary operator

A binary operator operates on two operands.

```
operand1 operator operand2
```

If an operator is given a null operand, the result is always null. The only operator that does not follow this rule is concatenation (||).

Arithmetic operators

You can use an arithmetic operator in an expression to negate, add, subtract, multiply, and divide numeric values. The result of this operation is also a numeric value. The + and - operators are also supported in date/time fields to allow date arithmetic. The following table lists the supported arithmetic operators.

Table 7: Arithmetic Operators

Operator	Purpose	Example
+ -	Denotes a positive or negative expression. These are unary operators.	`SELECT * FROM emp WHERE comm = -1`

Operator	Purpose	Example
* /	Multiplies, divides. These are binary operators.	`UPDATE emp SET sal = sal + sal * 0.10`
+ -	Adds, subtracts. These are binary operators.	`SELECT sal + comm FROM emp WHERE empno > 100`

Concatenation operator

The concatenation operator manipulates character strings. The following table lists the only supported concatenation operator.

Table 8: Concatenation Operator

Operator	Purpose	Example
||	Concatenates character strings.	`SELECT 'Name is' || ename FROM emp`

The result of concatenating two character strings is the data type VARCHAR.

Comparison operators

Comparison operators compare one expression to another. The result of such a comparison can be TRUE, FALSE, or UNKNOWN (if one of the operands is NULL). The driver considers the UNKNOWN result as FALSE.

The following table lists the supported comparison operators.

Table 9: Comparison Operators

Operator	Purpose	Example
=	Equality test.	`SELECT * FROM emp WHERE sal = 1500`
!=<>	Inequality test.	`SELECT * FROM emp WHERE sal != 1500`
><	"Greater than" and "less than" tests.	`SELECT * FROM emp WHERE sal > 1500 SELECT * FROM emp WHERE sal < 1500`
>=<=	"Greater than or equal to" and "less than or equal to" tests.	`SELECT * FROM emp WHERE sal >= 1500 SELECT * FROM emp WHERE sal <= 1500`

Operator	Purpose	Example
LIKE	% and _ wildcards can be used to search for a pattern in a column. The percent sign denotes zero, one, or multiple characters, while the underscore denotes a single character. The right-hand side of a LIKE expression must evaluate to a string or binary.	`SELECT * FROM emp WHERE ENAME LIKE 'J%'`

ESCAPE clause in LIKE operator
LIKE 'pattern string' ESCAPE 'c'
	The Escape clause is supported in the LIKE predicate to indicate the escape character. Escape characters are used in the pattern string to indicate that any wildcard character that is after the escape character in the pattern string should be treated as a regular character.
	The default escape character is backslash (\).
		`SELECT * FROM emp WHERE ENAME LIKE 'J%\_%' ESCAPE '\'`
		This matches all records with names that start with letter 'J' and have the '_' character in them.
		`SELECT * FROM emp WHERE ENAME LIKE 'JOE\_JOHN' ESCAPE '\'`
		This matches only records with name 'JOE_JOHN'.

[NOT] IN	"Equal to any member of" test.	`SELECT * FROM emp WHERE job IN ('CLERK','ANALYST') SELECT * FROM emp WHERE sal IN (SELECT sal FROM emp WHERE deptno = 30)`
[NOT] BETWEEN x AND y	"Greater than or equal to x" and "less than or equal to y."	`SELECT * FROM emp WHERE sal BETWEEN 2000 AND 3000`
EXISTS	Tests for existence of rows in a subquery.	`SELECT empno, ename, deptno FROM emp e WHERE EXISTS (SELECT deptno FROM dept WHERE e.deptno = dept.deptno)`
IS [NOT] NULL	Tests whether the value of the column or expression is NULL.	`SELECT * FROM emp WHERE ename IS NOT NULL SELECT * FROM emp WHERE ename IS NULL`

Logical operators

A logical operator combines the results of two component conditions to produce a single result or to invert the result of a single condition. The following table lists the supported logical operators.

Table 10: Logical Operators

Operator	Purpose	Example
NOT	Returns TRUE if the following condition is FALSE. Returns FALSE if it is TRUE. If it is UNKNOWN, it remains UNKNOWN.	`SELECT * FROM emp WHERE NOT (job IS NULL)` `SELECT * FROM emp WHERE NOT (sal BETWEEN 1000 AND 2000)`
AND	Returns TRUE if both component conditions are TRUE. Returns FALSE if either is FALSE; otherwise, returns UNKNOWN.	`SELECT * FROM emp WHERE job = 'CLERK' AND deptno = 10`
OR	Returns TRUE if either component condition is TRUE. Returns FALSE if both are FALSE; otherwise, returns UNKNOWN.	`SELECT * FROM emp WHERE job = 'CLERK' OR deptno = 10`

Example

In the Where clause of the following Select statement, the AND logical operator is used to ensure that managers earning more than $1000 a month are returned in the result:

```
SELECT * FROM emp WHERE jobtitle = manager AND sal > 1000
```

Operator precedence

As expressions become more complex, the order in which the expressions are evaluated becomes important. The following table shows the order in which the operators are evaluated. The operators in the first line are evaluated first, then those in the second line, and so on. Operators in the same line are evaluated left to right in the expression.You can change the order of precedence by using parentheses. Enclosing expressions in parentheses forces them to be evaluated together.

Table 11: Operator Precedence

Precedence	Operator
1	+ (Positive), - (Negative)
2	*(Multiply), / (Division)
3	+ (Add), - (Subtract)
4	|| (Concatenate)
5	=, >, <, >=, <=, <>, != (Comparison operators)
6	NOT, IN, LIKE
7	AND
8	OR

Example A

The query in the following example returns employee records for which the department number is 1 or 2 and the salary is greater than $1000:

```
SELECT * FROM emp WHERE (deptno = 1 OR deptno = 2) AND sal > 1000
```

Because parenthetical expressions are forced to be evaluated first, the OR operation takes precedence over AND.

Example B

In the following example, the query returns records for all the employees in department 1, but only employees whose salary is greater than $1000 in department 2.

```
SELECT * FROM emp WHERE deptno = 1 OR deptno = 2 AND sal > 1000
```

The AND operator takes precedence over OR, so that the search condition in the example is equivalent to the expression `deptno = 1 OR (deptno = 2 AND sal > 1000)`.

Functions

The driver supports a number of functions that you can use in expressions, including String, Numeric, Timedate, and System functions.

Refer to "Scalar functions" in the *Progress DataDirect for JDBC Drivers Reference* for more information.

Conditions

A condition specifies a combination of one or more expressions and logical operators that evaluates to either TRUE, FALSE, or UNKNOWN. You can use a condition in the Where clause of the Delete, Select, and Update statements; and in the Having clauses of Select statements. The following describes supported conditions.

Table 12: Conditions

Condition	Description
Simple comparison	Specifies a comparison with expressions or subquery results. `= , !=, <>, < , >, <=, <=`
Group comparison	Specifies a comparison with any or all members in a list or subquery. `[= , !=, <>, < , >, <=, <=] [ANY, ALL, SOME]`
Membership	Tests for membership in a list or subquery. `[NOT] IN`
Range	Tests for inclusion in a range. `[NOT] BETWEEN`

Condition	Description
NULL	Tests for nulls. `IS NULL, IS NOT NULL`
EXISTS	Tests for existence of rows in a subquery. `[NOT] EXISTS`
LIKE	Specifies a test involving pattern matching. `[NOT] LIKE`
Compound	Specifies a combination of other conditions. `CONDITION [AND/OR] CONDITION`

Subqueries

A query is an operation that retrieves data from one or more tables or views. In this reference, a top-level query is called a Select statement, and a query nested within a Select statement is called a subquery.

A subquery is a query expression that appears in the body of another expression such as a Select, an Update, or a Delete statement. In the following example, the second Select statement is a subquery:

```
SELECT * FROM emp WHERE deptno IN (SELECT deptno FROM dept)
```

IN predicate

Purpose

The In predicate specifies a set of values against which to compare a result set. If the values are being compared against a subquery, only a single column result set is returned.

Syntax

value [NOT] IN (*value1*, *value2*,...)

OR

value [NOT] IN (*subquery*)

Example

```
SELECT * FROM emp WHERE deptno IN
(SELECT deptno FROM dept WHERE dname <> 'Sales')
```

EXISTS predicate

Purpose

The Exists predicate is true only if the cardinality of the subquery is greater than 0; otherwise, it is false.

Syntax

```
EXISTS (subquery)
```

Example

```
SELECT empno, ename, deptno FROM emp e WHERE EXISTS
(SELECT deptno FROM dept WHERE e.deptno = dept.deptno)
```

UNIQUE predicate

Purpose

The Unique predicate is used to determine whether duplicate rows exist in a virtual table (one returned from a subquery).

Syntax

```
UNIQUE (subquery)
```

Example

```
SELECT * FROM dept d WHERE UNIQUE
(SELECT deptno FROM emp e WHERE e.deptno = d.deptno)
```

Correlated subqueries

Purpose

A correlated subquery is a subquery that references a column from a table referred to in the parent statement. A correlated subquery is evaluated once for each row processed by the parent statement. The parent statement can be a Select, Update, or Delete statement.

A correlated subquery answers a multiple-part question in which the answer depends on the value in each row processed by the parent statement. For example, you can use a correlated subquery to determine which employees earn more than the average salaries for their departments. In this case, the correlated subquery specifically computes the average salary for each department.

Syntax

```
SELECT select_list
   FROM table1 t_alias1
   WHERE expr rel_operator
   (SELECT column_list
       FROM table2 t_alias2
```

```
       WHERE t_alias1.columnrel_operatort_alias2.column)
UPDATE table1 t_alias1
   SET column =
   (SELECT expr
       FROM table2 t_alias2
        WHERE t_alias1.column = t_alias2.column)
DELETE FROM table1 t_alias1
   WHERE column rel_operator
   (SELECT expr
       FROM table2 t_alias2
       WHERE t_alias1.column = t_alias2.column)
```

Notes

- Correlated column names in correlated subqueries must be explicitly qualified with the table name of the parent.

Example A

The following statement returns data about employees whose salaries exceed their department average. This statement assigns an alias to emp, the table containing the salary information, and then uses the alias in a correlated subquery:

```
SELECT deptno, ename, sal FROM emp x WHERE sal >
  (SELECT AVG(sal) FROM emp WHERE x.deptno = deptno)
  ORDER BY deptno
```

Example B

This is an example of a correlated subquery that returns row values:

```
SELECT * FROM dept "outer" WHERE 'manager' IN
  (SELECT managername FROM emp
  WHERE "outer".deptno = emp.deptno)
```

Example C

This is an example of finding the department number (deptno) with multiple employees:

```
SELECT * FROM dept main WHERE 1 <
  (SELECT COUNT(*) FROM emp WHERE deptno = main.deptno)
```

Example D

This is an example of correlating a table with itself:

```
SELECT deptno, ename, sal FROM emp x WHERE sal >
  (SELECT AVG(sal) FROM emp WHERE x.deptno = deptno)
```

6

Introduction to the HubSpot Data Model

The HubSpot data model is defined using a collection of standard JSON documents that contain the data, identifiers, and object relationships for a given service. These documents are stored on URL endpoints that are accessible using sets of proprietary REST API calls. To expose HubSpot resources to SQL applications, the driver maps HubSpot endpoints to a set of relational parent and child tables. The following sections describe the relational tables exposed by the driver along with their corresponding HubSpot REST API call.

For details, see the following topics:

- CAMPAIGNS
- COMMENTS
- COMPANIES
- COMPANIES_V3
- CONTACT
- CONTACTIDENTITYPROFILEIDENTITIES
- CONTACTIDENTITYPROFILES
- CONTACTLISTIDENTITYPROFILEIDENTITIES
- CONTACTLISTIDENTITYPROFILES
- CONTACTLISTS
- CONTACTS
- CONTACTSINLIST
- CONTACTS_V3
- CRMASSOCIATIONS
- DEALASSOCIATEDCOMPANYIDS
- DEALASSOCIATEDCONTACTIDS
- DEALASSOCIATEDDEALIDS
- DEALASSOCIATEDTICKETIDS
- DEALASSOCIATIONS
- DEALPIPELINES
- DEALPIPELINESTAGES
- DEALS
- DEALS_V3
- DOMAINS
- ECOMMERCESYNCERRORS
- EMAILSUBSCRIPTIONS
- ENGAGEMENTASSOCIATEDCOMPANIES
- ENGAGEMENTASSOCIATEDCONTACTIDS
- ENGAGEMENTASSOCIATEDCONTENTIDS
- ENGAGEMENTASSOCIATEDDEALIDS
- ENGAGEMENTASSOCIATEDOWNERIDS
- ENGAGEMENTASSOCIATEDQUOTEIDS
- ENGAGEMENTASSOCIATEDTICKETIDS

- ENGAGEMENTASSOCIATEDWORKFLOWIDS
- ENGAGEMENTS
- ENGAGEMENTSSCHEDULEDTASKS
- EVENTS
- FILES
- FOLDERS
- FORMFIELDGROUPS
- FORMFIELDOPTIONS
- FORMFIELDS
- FORMFIELDSELECTEDOPTIONS
- FORMS
- LINEITEMS
- LINEITEMS_V3
- MARKETINGEMAILS
- OWNERS
- OWNERSREMOTELISTS
- PAGES
- PRODUCTS
- PRODUCTS_V3
- QUOTES_V3
- SOCIALMEDIACHANNELS
- SOCIALMEDIAMESSAGES
- TASKS
- TEMPLATES
- TICKETPIPELINES
- TICKETPIPELINESTAGES
- TICKETS
- TICKETS_V3
- URLMAPPINGS
- WORKFLOW

ALLEMAILCAMPAIGNIDS

Endpoint

```
https://api.hubapi.com/marketing-emails/v1/emails
```

Parent Table

MARKETINGEMAILS

Columns

The ALLEMAILCAMPAIGNIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
MARKETINGEMAILS_ID*	BigInt	References: MARKETINGEMAILS.ID
POSITION*	Integer	
ALLEMAILCAMPAIGNID	Integer	

ANALYTICS

Endpoint

```
https://api.hubapi.com/analytics/v2/reports/{BreakdownBy}/{TimePeriod}
```

Columns

The ANALYTICS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
BREAKDOWNBY*	VarChar(64)
TIMEPERIOD*	VarChar(64)
STARTDATE*	VarChar(64)
ENDDATE*	VarChar(64)
BOUNCERATE	Double
CONTACTS	Integer
CONTACTSPERPAGEVIEW	Double

Column Name	Data Type
CONTACTTOCUSTOMERRATE	Double
CUSTOMERS	Integer
LEADS	Integer
LEADSPERVIEW	Double
MARKETINGQUALIFIEDLEADS	Integer
NEWVISITORSESSIONRATE	Double
OPPORTUNITIES	Integer
PAGEVIEWSMINUSEXITS	Integer
PAGEVIEWSPERSESSION	Double
RAWVIEWS	Integer
RETURNINGVISITS	Integer
SALESQUALIFIEDLEADS	Integer
SESSIONTOCONTACTRATE	Double
STANDARDVIEWS	Integer
SUBSCRIBERS	Integer
TIMEPERSESSION	Double
TOTAL	Integer
VISITORS	Integer
VISITS	Integer

ANALYTICSBREAKDOWNS

Endpoint

`https://api.hubapi.com/analytics/v2/reports/{BreakdownBy}/{TimePeriod}`

Parent Table

ANALYTICS

Columns

The ANALYTICSBREAKDOWNS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ANALYTICS_BREAKDOWNBY*	VarChar(64)	References: ANALYTICS.BREAKDOWNBY
ANALYTICS_TIMEPERIOD*	VarChar(64)	References: ANALYTICS.TIMEPERIOD
ANALYTICS_STARTDATE*	VarChar(64)	References: ANALYTICS.STARTDATE
ANALYTICS_ENDDATE*	VarChar(64)	References: ANALYTICS.ENDDATE
POSITION*	Integer	
BOUNCERATE	Double	
BREAKDOWN	VarChar(64)	
CONTACTS	Integer	
CONTACTSPERPAGEVIEW	Double	
CONTACTTOCUSTOMERRATE	Double	
CUSTOMERS	Integer	
LEADS	Integer	
LEADSPERVIEW	Double	
MAPPEDBREAKDOWN	Integer	
MARKETINGQUALIFIEDLEADS	Integer	
NEWVISITORSESSIONRATE	Double	
OPPORTUNITIES	Integer	
PAGEVIEWSMINUSEXITS	Integer	
PAGEVIEWSPERSESSION	Double	
RAWVIEWS	Integer	
RETURNINGVISITS	Integer	
SALESQUALIFIEDLEADS	Integer	
SESSIONTOCONTACTRATE	Double	
STANDARDVIEWS	Integer	

Column Name	Data Type	Notes
SUBSCRIBERS	Integer	
TIMEPERSESSION	Double	
VISITORS	Integer	
VISITS	Integer	

ANALYTICSCONTENTBREAKDOWNS

Endpoint

`https://api.hubapi.com/analytics/v2/reports/{ContentType}/{TimePeriod}`

Parent Table

ANALYTICSCONTENTS

Columns

The ANALYTICSCONTENTBREAKDOWNS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ANALYTICSCONTENTS_CONTENTTYPE*	VarChar(64)	References: ANALYTICSCONTENTS .CONTENTTYPE
ANALYTICSCONTENTS_TIMEPERIOD*	VarChar(64)	References: ANALYTICSCONTENTS.TIMEPERIOD
ANALYTICSCONTENTS_STARTDATE*	VarChar(64)	References: ANALYTICSCONTENTS.STARTDATE
ANALYTICSCONTENTS_ENDDATE*	VarChar(64)	References: ANALYTICSCONTENTS.ENDDATE
POSITION*	Integer	
BREAKDOWN	BigInt	
CONTACTS	Integer	
CTAVIEWS	Integer	
ENTRANCES	Integer	
EXITS	Integer	
EXITSPERPAGEVIEW	Double	
LEADS	Integer	

Column Name	Data Type	Notes
MAPPEDBREAKDOWN	Integer	
PAGEBOUNCERATE	Double	
PAGETIME	Integer	
PAGEVIEWSMINUSEXITS	Integer	
RAWVIEWS	Integer	
STANDARDVIEWS	Integer	
SUBMISSIONS	Integer	
TIMEPERPAGEVIEW	Double	

ANALYTICSCONTENTS

Endpoint

```
https://api.hubapi.com/analytics/v2/reports/{ContentType}/{TimePeriod}
```

Columns

The ANALYTICSCONTENTS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
CONTENTTYPE*	VarChar(64)
TIMEPERIOD*	VarChar(64)
STARTDATE*	VarChar(64)
ENDDATE*	VarChar(64)
CONTACTS	Integer
CTAVIEWS	Integer
ENTRANCES	Integer
EXITS	Integer
EXITSPERPAGEVIEW	Double
LEADS	Integer

Column Name	Data Type
PAGEBOUNCERATE	Double
PAGETIME	Integer
PAGEVIEWSMINUSEXITS	Integer
RAWVIEWS	Integer
STANDARDVIEWS	Integer
SUBMISSIONS	Integer
TIMEPERPAGEVIEW	Double
TOTAL	Integer

ANALYTICSSPECIFICOBJECT

Endpoint

```
https://api.hubapi.com/analytics/v2/reports/{ObjectType}/{TimePeriod}
```

Columns

The ANALYTICSSPECIFICOBJECT table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
OBJECTTYPE*	VarChar(64)
TIMEPERIOD*	VarChar(64)
STARTDATE*	VarChar(64)
ENDDATE*	VarChar(64)
CTAVIEWS	Integer
ENTRANCES	Integer
EXITS	Integer
EXITSPERPAGEVIEW	Double
FORMVIEWS	Integer
INTERACTIONS	Integer

Column Name	Data Type
PAGEBOUNCERATE	Double
RAWVIEWS	Integer
TIMEPERPAGEVIEW	Double
TOTAL	Integer
VISIBLES	Integer

ANALYTICSSPECIFICOBJECTBREAKDOWNS

Endpoint

```
https://api.hubapi.com/analytics/v2/reports/{ObjectType}/{TimePeriod}
```

Parent Table

ANALYTICSSPECIFICOBJECT

Columns

The ANALYTICSSPECIFICOBJECTBREAKDOWNS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ANALYTICSSPECIFICOBJECT_OBJECTTYPE*	VarChar(64)	References: ANALYTICSSPECIFICOBJECT .OBJECTTYPE
ANALYTICSSPECIFICOBJECT_TIMEPERIOD*	VarChar(64)	References: ANALYTICSSPECIFICOBJECT .TIMEPERIOD
ANALYTICSSPECIFICOBJECT_STARTDATE*	VarChar(64)	References: ANALYTICSSPECIFICOBJECT .STARTDATE
ANALYTICSSPECIFICOBJECT_ENDDATE*	VarChar(64)	References: ANALYTICSSPECIFICOBJECT .ENDDATE
POSITION*	Integer	
BREAKDOWN	GUID	
CTAVIEWS	Integer	
ENTRANCES	Integer	
EXITS	Integer	

Column Name	Data Type	Notes
EXITSPERPAGEVIEW	Double	
FORMVIEWS	Integer	
INTERACTIONS	Integer	
MAPPEDBREAKDOWN	Integer	
PAGEBOUNCERATE	Double	
RAWVIEWS	Integer	
TIMEPERPAGEVIEW	Double	
VISIBLES	Integer	

BLOGAUTHORS

Endpoint

`https://api.hubapi.com/blogs/v3/blog-authors`

Columns

The BLOGAUTHORS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
CREATED	Timestamp(3)
DELETEDAT	Timestamp(3)
DISPLAYNAME	VarChar(64)
EMAIL	VarChar(64)
FACEBOOK	VarChar(64)
FULLNAME	VarChar(64)
HASSOCIALPROFILES	Boolean
LINKEDIN	VarChar(64)
PORTALID	Integer

Column Name	Data Type
SLUG	VarChar(64)
TWITTER	VarChar(64)
UPDATED	Timestamp(3)
USERID	Integer
USERNAME	VarChar(64)
WEBSITE	VarChar(64)

BLOGPOSTS

Endpoint

`https://api.hubapi.com/content/api/v2/blog-posts`

Columns

The BLOGPOSTS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
AUTHORNAME	VarChar(64)
AUTHORUSERID	Integer
BLOGAUTHORID	BigInt
CONTENTGROUPID	BigInt
CREATEDAT	Timestamp(3)
DELETEDAT	Timestamp(3)
NAME	VarChar(64)
POSTSUMMARY	VarChar(64)
PUBLISHEDAT	Timestamp(3)
SLUG	VarChar(64)

Column Name	Data Type
UPDATEDAT	Timestamp(3)
URL	VarChar(94)

BLOGS

Endpoint

`https://api.hubapi.com/content/api/v2/blogs`

Columns

The BLOGS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
ALLOWCOMMENTS	Boolean
COMMENTSHOULDCREATECONTACT	Boolean
CREATEDAT	Timestamp(3)
HTMLTITLE	VarChar(64)
LANGUAGE	VarChar(64)
NAME	VarChar(64)
POSTSPERLISTINGPAGE	Integer
POSTSPERRSSFEED	Integer
PUBLICTITLE	VarChar(64)
ROOTURL	VarChar(76)
SHOWSOCIALLINKFACEBOOK	Boolean
SHOWSOCIALLINKLINKEDIN	Boolean
SHOWSOCIALLINKTWITTER	Boolean
SLUG	VarChar(64)
UPDATEDAT	Timestamp(3)

BLOGTOPICIDS

Endpoint

`https://api.hubapi.com/content/api/v2/blog-posts`

Parent Table

BLOGPOSTS

Columns

The BLOGTOPICIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
BLOGPOSTS_ID*	BigInt	References: BLOGPOSTS.ID
POSITION*	Integer	
BLOGTOPICID	BigInt	

BLOGTOPICS

Endpoint

`https://api.hubapi.com/blogs/v3/topics`

Columns

The BLOGTOPICS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
CREATED	Timestamp(3)
DELETEDAT	Timestamp(3)
DESCRIPTION	VarChar(64)
NAME	VarChar(64)
PORTALID	Integer

Column Name	Data Type
SLUG	VarChar(64)
UPDATED	Timestamp(3)

CAMPAIGNINFO

Endpoint

```
https://api.hubapi.com/email/public/v1/campaigns/{CampaignId}
```

Columns

The CAMPAIGNINFO table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
CAMPAIGNID*	Integer
APPID	Integer
APPNAME	VarChar(64)
CONTENTID	BigInt
DELIVERED	Integer
ID	Integer
NAME	VarChar(128)
NUMINCLUDED	Integer
NUMQUEUED	Integer
OPEN	Integer
PROCESSED	Integer
SENT	Integer
SUBJECT	VarChar(64)
SUBTYPE	VarChar(64)
TYPE	VarChar(64)

CAMPAIGNS

Endpoint

```
https://api.hubapi.com/email/public/v1/campaigns/by-id
```

Columns

The CAMPAIGNS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
APPID	Integer
APPNAME	VarChar(64)

COMMENTS

Endpoint

```
https://api.hubapi.com/comments/v3/comments
```

Columns

The COMMENTS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
COLLECTIONID	BigInt
COMMENT	VarChar(64)
CONTENTAUTHOREMAIL	VarChar(64)
CONTENTAUTHORNAME	VarChar(64)
CONTENTCREATEDAT	BigInt
CONTENTID	BigInt
CONTENTPERMALINK	VarChar(94)
CONTENTTITLE	VarChar(64)

Column Name	Data Type
CREATEDAT	Timestamp(3)
DELETEDAT	Timestamp(3)
EXTRACONTEXT	VarChar(64)
FIRSTNAME	VarChar(64)
LASTNAME	VarChar(64)
LEGACYID	Integer
PARENTID	Integer
PORTALID	Integer
POSTID	Integer
REPLYINGTO	VarChar(64)
STATE	VarChar(64)
THREADID	VarChar(73)
USERAGENT	VarChar(64)
USEREMAIL	VarChar(64)
USERIP	VarChar(64)
USERNAME	VarChar(64)
USERREFERRER	VarChar(64)
USERURL	VarChar(64)

COMPANIES

Endpoint

```
https://api.hubapi.com/companies/v2/companies/{CompanyId}
```

Columns

The COMPANIES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
COMPANYID*	BigInt
ALLOWNERIDS	Integer
ANALYTICSFIRSTTIMESTAMP	Date
ANALYTICSNOOFPAGEVIEWS	Integer
ANALYTICSSOURCE	VarChar(64)
ANALYTICSSOURCEDATA1	VarChar(64)
ANALYTICSSOURCEDATA2	VarChar(64)
ANNUALREVENUE	BigInt
CITY	VarChar(64)
CLOSEDATE	Date
DATECREATED	Date
DAYSTOCLOSE	Integer
DESCRIPTION	VarChar(64)
DOMAIN	VarChar(64)
FILEMANAGERKEY	VarChar(82)
FIRSTCONTACTCREATEDATE	Date
FIRSTDEALCREATEDDATE	Date
HUBSPOTOWNERASSIGNDATE	Date
HUBSPOTOWNERID	Integer
INDUSTRY	VarChar(64)
ISDELETED	Boolean
LASTCONTACTED	Date
LASTMODIFIEDDATE	Date
LASTUPDATED	Date
LIFECYCLESTAGE	VarChar(64)
LINKEDPAGELINK	VarChar(64)

Column Name	Data Type
NAME	VarChar(64)
NOOFANALYTICSVISITS	Integer
NOOFASSOCIATEDCONTACTS	Integer
NOOFASSOCIATEDDEALS	Integer
NOOFCHILDREN	Integer
NOOFCONTACTEDNOTES	Integer
NOOFEMPLOYEES	Integer
NOOFNOTES	Integer
NOTESNEXTACTIVITYDATE	Date
OBJECTID	BigInt
PARENTCOMPANYID	BigInt
PHONENO	VarChar(64)
PORTALID	Integer
PREDICTIVECONTACTSSCORE	Double
RECENTDEALAMOUNT	Integer
RECENTDEALCLOSEDATE	Date
STATE	VarChar(64)
TIMEZONE	VarChar(64)
TOTALREVENUE	Integer
TYPE	VarChar(64)
WEBSITE	VarChar(64)
YEARFOUNDED	Integer
ZIP	VarChar(64)

COMPANIES_V3

Endpoint

```
https://api.hubapi.com/crm/v3/objects/companies
```

Columns

The COMPANIES_V3 table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
ARCHIVED	Boolean
ASSOCIATEDCONTACTS	VarChar(64)
ASSOCIATEDDEALS	VarChar(64)
CONTACTEDNOTES	VarChar(64)
CREATEDAT	Timestamp(9)
FACEBOOKCOMPANYPAGE	VarChar(82)
FIRSTCONTACTCREATEDATE	VarChar(64)
FIRSTDEALCREATEDDATE	VarChar(64)
HSANALYTICSFIRSTTIMESTAMP	VarChar(64)
HSANALYTICSNUMPAGEVIEWS	VarChar(64)
HSANALYTICSNUMVISITS	VarChar(64)
HSANALYTICSSOURCE	VarChar(64)
HSANALYTICSSOURCEDATA1	VarChar(64)
HSANALYTICSSOURCEDATA2	VarChar(64)
HSPREDICTIVECONTACTSCOREV2	VarChar(64)
ISPUBLIC	VarChar(64)
LASTCONTACTED	VarChar(64)
LINKEDINCOMPANYPAGE	VarChar(123)

Column Name	Data Type
NOTES	VarChar(64)
PROPERTIES_ADDRESS	VarChar(64)
PROPERTIES_ANNUALREVENUE	Integer
PROPERTIES_CITY	VarChar(64)
PROPERTIES_COUNTRY	VarChar(64)
PROPERTIES_DESCRIPTION	VarChar(423)
PROPERTIES_DOMAIN	VarChar(64)
PROPERTIES_INDUSTRY	VarChar(64)
PROPERTIES_LIFECYCLESTAGE	VarChar(64)
PROPERTIES_LINKEDINBIO	VarChar(423)
PROPERTIES_NAME	VarChar(64)
PROPERTIES_NOTES_LAST_UPDATED	VarChar(64)
PROPERTIES_NOTES_NEXT_ACTIVITY_DATE	VarChar(64)
PROPERTIES_NUMBEROFEMPLOYEES	Integer
PROPERTIES_PHONE	VarChar(64)
PROPERTIES_STATE	VarChar(64)
PROPERTIES_TIMEZONE	VarChar(64)
PROPERTIES_TWITTERHANDLE	VarChar(64)
PROPERTIES_WEB_TECHNOLOGIES	VarChar(417)
PROPERTIES_WEBSITE	VarChar(64)
PROPERTIES_ZIP	VarChar(64)
UPDATEDAT	Timestamp(9)
YEARFOUNDED	Integer

CONTACT

Endpoint

```
https://api.hubapi.com/contacts/v1/contact/vid/{vid}/profile
```

Columns

The CONTACT table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
VID*	Integer
ADDRESS	VarChar(64)
CALCULATEDPHONENO	VarChar(64)
CALCULATEDPHONENOCOUNTRYCODE	VarChar(64)
CITY	VarChar(64)
DATECREATED	Timestamp(3)
EMAIL	VarChar(64)
FIRSTNAME	VarChar(64)
HSANALYTICSAVERAGEPAGEVIEWS	Integer
HSANALYTICSFIRSTTIMESTAMP	BigInt
HSANALYTICSFIRSTTOUCHCONVERTINGCAMPAIGN	VarChar(64)
HSANALYTICSFIRSTVISITTIMESTAMP	VarChar(64)
HSANALYTICSLASTTIMESTAMP	VarChar(64)
HSANALYTICSLASTTOUCHCONVERTINGCAMPAIGN	VarChar(64)
HSANALYTICSLASTVISITTIMESTAMP	VarChar(64)
HSANALYTICSNUMEVENTCOMPLETIONS	Integer
HSANALYTICSNUMPAGEVIEWS	Integer
HSANALYTICSNUMVISITS	Integer
HSANALYTICSREVENUE	Double

Column Name	Data Type
HSANALYTICSSOURCE	VarChar(64)
HSANALYTICSSOURCEDATA1	VarChar(64)
HSANALYTICSSOURCEDATA2	Integer
HSEMAILDOMAIN	VarChar(64)
HSEMAILOPTOUT	VarChar(64)
HSEMAILOPTOUT6841412	VarChar(64)
HSEMAILOPTOUT6841413	VarChar(64)
HSEMAILOPTOUT6841782	VarChar(64)
HSEMAILOPTOUT6841830	VarChar(64)
HSEMAILQUARANTINED	VarChar(64)
HSISCONTACT	VarChar(64)
HSLIFECYCLESTAGESUBSCRIBERDATE	Date
HSOBJECTID	Integer
HSPREDICTIVECONTACTSCOREV2	Double
HSPREDICTIVESCORINGTIER	VarChar(64)
HSSEARCHABLECALCULATEDPHONENUMBER	BigInt
HSSOCIALFACEBOOKCLICKS	Integer
HSSOCIALGOOGLEPLUSCLICKS	Integer
HSSOCIALLASTENGAGEMENT	VarChar(64)
HSSOCIALLINKEDINCLICKS	Integer
HSSOCIALNUMBROADCASTCLICKS	VarChar(64)
HSSOCIALTWITTERCLICKS	Integer
ISCONTACT	Boolean
LASTMODIFIEDDATE	Timestamp(3)
LASTNAME	VarChar(64)
LIFECYCLESTAGE	VarChar(64)

Column Name	Data Type
NUMCONVERSIONEVENTS	Integer
NUMUNIQUECONVERSIONEVENTS	Integer
PHONENO	VarChar(64)
PORTALID	Integer
PROFILEURL	VarChar(78)
PROPERTIES_HS_SOCIAL_NUM_BROADCAST_CLICKS_VALUE	Integer
STATE	VarChar(64)
ZIP	Integer

CONTACTIDENTITYPROFILEIDENTITIES

Endpoint

`https://api.hubapi.com/contacts/v1/lists/all/contacts/all`

Parent Table

CONTACTIDENTITYPROFILES

Columns

The CONTACTIDENTITYPROFILEIDENTITIES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
CONTACTS_VID*	Integer	References: CONTACTS.VID
CONTACTIDENTITYPROFILES_POSITION*	Integer	References: CONTACTIDENTITYPROFILES .POSITION
POSITION*	Integer	
ISPRIMARY	Boolean	
TIMESTAMP	Timestamp(3)	
TYPE	VarChar(13)	
VALUE	VarChar(64)	

CONTACTIDENTITYPROFILES

Endpoint

`https://api.hubapi.com/contacts/v1/lists/all/contacts/all`

Parent Table

CONTACTS

Columns

The CONTACTIDENTITYPROFILES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
CONTACTS_VID*	Integer	References: CONTACTS.VID
POSITION*	Integer	
CONTACTVID	Integer	
DELETEDCHANGEDTIMESTAMP	Integer	
SAVEDAT	Timestamp(3)	

CONTACTLISTIDENTITYPROFILEIDENTITIES

Endpoint

`https://api.hubapi.com/contacts/v1/lists/{ListId}/contacts/all`

Parent Table

CONTACTLISTIDENTITYPROFILES

Columns

The CONTACTLISTIDENTITYPROFILEIDENTITIES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
CONTACTSINLIST_LISTID*	Integer	References: CONTACTSINLIST.LISTID
CONTACTLISTIDENTITYPROFILES _POSITION*	Integer	References: CONTACTLISTIDENTITYPROFILES .POSITION
POSITION*	Integer	

Column Name	Data Type	Notes
ISPRIMARY	Boolean	
TIMESTAMP	BigInt	
TYPE	VarChar(13)	
VALUE	VarChar(64)	

CONTACTLISTIDENTITYPROFILES

Endpoint

`https://api.hubapi.com/contacts/v1/lists/{ListId}/contacts/all`

Parent Table

CONTACTSINLIST

Columns

The CONTACTLISTIDENTITYPROFILES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
CONTACTSINLIST_LISTID*	Integer	References: CONTACTSINLIST.LISTID
POSITION*	Integer	
DELETEDCHANGEDTIMESTAMP	Integer	
SAVEDATTIMESTAMP	BigInt	
VID	Integer	

CONTACTLISTS

Endpoint

`https://api.hubapi.com/contacts/v1/lists`

Columns

The CONTACTLISTS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PORTALID*	Integer
CREATEDAT	Timestamp(3)
FILTERSAGGREGATE	JSON(16777215)
INTERNALLISTID	Integer
ISDYNAMIC	Boolean
LASTPROCESSINGSTATECHANGEAT	Timestamp(3)
LASTSIZECHANGEAT	Timestamp(3)
LISTID	Integer
LISTSIZE	Integer
NAME	VarChar(64)
PROCESSINGSTATE	VarChar(64)
UPDATEDAT	Timestamp(3)

CONTACTS

Endpoint

`https://api.hubapi.com/contacts/v1/lists/all/contacts/all`

Columns

The CONTACTS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
VID*	Integer
ADDEDAT	Timestamp(3)
COMPANY	VarChar(64)
FIRSTNAME	VarChar(64)
ISCONTACT	Boolean
LASTMODFIFIEDAT	Timestamp(3)

Column Name	Data Type
LASTNAME	VarChar(64)
PORTALID	Integer
PROFILEURL	VarChar(78)

CONTACTSINLIST

Endpoint

```
https://api.hubapi.com/contacts/v1/lists/{ListId}/contacts/all
```

Columns

The CONTACTSINLIST table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
LISTID*	Integer
ADDEDAT	Timestamp(3)
FIRSTNAME	VarChar(64)
ISCONTACT	Boolean
LASTMODFIFIEDAT	Timestamp(3)
LASTNAME	VarChar(64)
PROFILEURL	VarChar(78)
VID	Integer

CONTACTS_V3

Endpoint

```
https://api.hubapi.com/crm/v3/objects/contacts
```

Columns

The CONTACTS_V3 table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
ARCHIVED	Boolean
CREATEDAT	Timestamp(9)
HSANALYTICSAVERAGEPAGEVIEWS	Integer
HSANALYTICSFIRSTTIMESTAMP	Timestamp(9)
HSANALYTICSFIRSTTOUCHCONVERTINGCAMPAIGN	VarChar(64)
HSANALYTICSFIRSTVISITTIMESTAMP	VarChar(64)
HSANALYTICSLASTTIMESTAMP	VarChar(64)
HSANALYTICSLASTTOUCHCONVERTINGCAMPAIGN	VarChar(64)
HSANALYTICSLASTVISITTIMESTAMP	VarChar(64)
HSANALYTICSNUMEVENTCOMPLETIONS	Integer
HSANALYTICSNUMPAGEVIEWS	Integer
HSANALYTICSNUMVISITS	Integer
HSANALYTICSREVENUE	Double
HSANALYTICSSOURCE	VarChar(64)
HSANALYTICSSOURCEDATA1	VarChar(64)
HSANALYTICSSOURCEDATA2	VarChar(64)
HSCALCULATEDPHONENUMBER	VarChar(64)
HSCALCULATEDPHONENUMBERCOUNTRYCODE	VarChar(64)
HSEMAILDOMAIN	VarChar(64)
HSEMAILOPTOUT	VarChar(64)
HSEMAILOPTOUT6841412	VarChar(64)
HSEMAILOPTOUT6841413	VarChar(64)
HSEMAILOPTOUT6841782	VarChar(64)
HSEMAILOPTOUT6841830	VarChar(64)
HSEMAILQUARANTINED	VarChar(64)

Column Name	Data Type
HSEMAILRECIPIENTFATIGUERECOVERYTIME	VarChar(64)
HSISCONTACT	VarChar(64)
HSLIFECYCLESTAGESUBSCRIBERDATE	Timestamp(9)
HSPREDICTIVECONTACTSCOREV2	Double
HSPREDICTIVESCORINGTIER	VarChar(64)
HSSEARCHABLECALCULATEDPHONENUMBER	BigInt
HSSOCIALFACEBOOKCLICKS	Integer
HSSOCIALGOOGLEPLUSCLICKS	Integer
HSSOCIALLASTENGAGEMENT	VarChar(64)
HSSOCIALLINKEDINCLICKS	Integer
HSSOCIALNUMBROADCASTCLICKS	Integer
HSSOCIALTWITTERCLICKS	Integer
NUMCONVERSIONEVENTS	Integer
NUMUNIQUECONVERSIONEVENTS	Integer
PROPERTIES_ADDRESS	VarChar(64)
PROPERTIES_ASSOCIATEDCOMPANYID	VarChar(64)
PROPERTIES_CITY	VarChar(64)
PROPERTIES_EMAIL	VarChar(64)
PROPERTIES_FIRSTNAME	VarChar(64)
PROPERTIES_LASTNAME	VarChar(64)
PROPERTIES_LIFECYCLESTAGE	VarChar(64)
PROPERTIES_PHONE	VarChar(64)
PROPERTIES_STATE	VarChar(64)
PROPERTIES_ZIP	Integer
UPDATEDAT	Timestamp(9)

CRMASSOCIATIONS

Endpoint

```
https://api.hubapi.com/crm-associations/v1/associations/{ObjectId}/HUBSPOT_DEFINED/{DefinitionId}
```

Columns

The CRMASSOCIATIONS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
OBJECTID*	BigInt
DEFINITIONID*	Integer
RESULTID	Integer

DEALASSOCIATEDCOMPANYIDS

Endpoint

```
https://api.hubapi.com/deals/v1/deal/{DealId}
```

Parent Table

DEALS

Columns

The DEALASSOCIATEDCOMPANYIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
DEALS_DEALID*	Integer	References: DEALS.DEALID
POSITION*	Integer	
DEALASSOCIATEDCOMPANYID	BigInt	

DEALASSOCIATEDCONTACTIDS

Endpoint

`https://api.hubapi.com/deals/v1/deal/{DealId}`

Parent Table

DEALS

Columns

The DEALASSOCIATEDCONTACTIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
DEALS_DEALID*	Integer	References: DEALS.DEALID
POSITION*	Integer	
DEALASSOCIATEDCONTACTID	BigInt	

DEALASSOCIATEDDEALIDS

Endpoint

`https://api.hubapi.com/deals/v1/deal/{DealId}`

Parent Table

DEALS

Columns

The DEALASSOCIATEDDEALIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
DEALS_DEALID*	Integer	References: DEALS.DEALID
POSITION*	Integer	
DEALASSOCIATEDDEALID	BigInt	

DEALASSOCIATEDTICKETIDS

Endpoint

`https://api.hubapi.com/deals/v1/deal/{DealId}`

Parent Table

DEALS

Columns

The DEALASSOCIATEDTICKETIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
DEALS_DEALID*	Integer	References: DEALS.DEALID
POSITION*	Integer	
DEALASSOCIATEDTICKETID	BigInt	

DEALASSOCIATIONS

Endpoint

`https://api.hubapi.com/deals/v1/deal/associated/{ObjectType}/{ObjectId}/paged`

Columns

The DEALASSOCIATIONS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
OBJECTTYPE*	VarChar(64)
DEALID	Integer
OBJECTID	BigInt
PORTALID	Integer

DEALPIPELINES

Endpoint

```
https://api.hubapi.com/crm-pipelines/v1/pipelines/deals
```

Columns

The DEALPIPELINES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PIPELINEID*	VarChar(64)
ACTIVE	Boolean
CREATEDAT	Integer
DEFAULT_	Boolean
DISPLAYORDER	Integer
LABEL	VarChar(64)
OBJECTTYPE	VarChar(64)
OBJECTTYPEID	VarChar(64)
UPDATEDAT	VarChar(64)

DEALPIPELINESTAGES

Endpoint

```
https://api.hubapi.com/crm-pipelines/v1/pipelines/deals
```

Parent Table

DEALPIPELINES

Columns

The DEALPIPELINESTAGES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
DEALPIPELINES_PIPELINEID*	VarChar(64)	References: DEALPIPELINES.PIPELINEID

Column Name	Data Type	Notes
POSITION*	Integer	
ACTIVE	Boolean	
CREATEDAT	Integer	
DISPLAYORDER	Integer	
ISCLOSED	Boolean	
PROBABILITY	Double	
STAGEID	VarChar(64)	
STAGENAME	VarChar(64)	
UPDATEDAT	VarChar(64)	

DEALS

Endpoint

`https://api.hubapi.com/deals/v1/deal/{DealId}`

Columns

The DEALS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
DEALID*	Integer
ALLACCESSIBLETEAMIDS	Integer
ALLOWNERIDS	Integer
ALLTEAMIDS	Integer
AMOUNT	Double
AMOUNTINHOMECURRENCY	Double
CLOSEDATE	Date
CLOSEDDEALAMOUNT	Integer
CLOSEDDEALAMOUNTINHOMECURRENCY	Integer

Column Name	Data Type
CREATEDATE	Date
DAYSTOCLOSE	Integer
DEALNAME	VarChar(64)
DEALSTAGEPROBABILITY	Integer
HUBSPOTCREATEDATE	Date
HUBSPOTOBJECTID	Integer
HUBSPOTOWNERID	Integer
HUBSPOTTEAMID	Integer
ISDELETED	Boolean
LASTMODIFIEDDATE	Date
NUMBEROFCONTACTS	Integer
OWNERASSIGNEDDATE	Date
PORTALID	Integer
PROJECTEDDEALAMOUNT	Double
PROJECTEDDEALAMOUNTINHOMECURRENCY	Double

DEALS_V3

Endpoint

`https://api.hubapi.com/crm/v3/objects/deals`

Columns

The DEALS_V3 table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
AMOUNTINHOMECURRENCY	Double
ARCHIVED	Boolean

Column Name	Data Type
CREATEDAT	Timestamp(9)
DAYSTOCLOSE	Integer
HSALLACCESSIBLETEAMIDS	Integer
HSALLOWNERIDS	Integer
HSALLTEAMIDS	Integer
HSCLOSEDAMOUNT	Integer
HSCLOSEDAMOUNTINHOMECURRENCY	Integer
HSCREATEDATE	Timestamp(9)
HSDEALSTAGEPROBABILITY	Integer
HSPROJECTEDAMOUNT	Double
HSPROJECTEDAMOUNTINHOMECURRENCY	Double
HUBSPOTOWNERASSIGNEDDATE	Timestamp(9)
HUBSPOTOWNERID	Integer
HUBSPOTTEAMID	Integer
NUMASSOCIATEDCONTACTS	Integer
PROPERTIES_AMOUNT	Double
PROPERTIES_CLOSEDATE	Timestamp(0)
PROPERTIES_DEALNAME	VarChar(64)
UPDATEDAT	Timestamp(9)

DOMAINS

Endpoint

`https://api.hubapi.com/cos-domains/v1/domains`

Columns

The DOMAINS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
CREATED	Timestamp(3)
DOMAIN	VarChar(64)
ISANYPRIMARY	Boolean
ISDNSCORRECT	Boolean
ISLEGACYDOMAIN	Boolean
ISRESOLVING	Boolean
MANUALLYMARKEDASRESOLVING	Boolean
PRIMARYBLOGPOST	Boolean
PRIMARYEMAIL	Boolean
PRIMARYLANDINGPAGE	Boolean
PRIMARYLEGACYPAGE	Boolean
PRIMARYSITEPAGE	Boolean
SECONDARYTODOMAIN	VarChar(64)
UPDATED	Timestamp(3)

ECOMMERCESYNCERRORS

Endpoint

`https://api.hubapi.com/extensions/ecomm/v2/sync/errors`

Columns

The ECOMMERCESYNCERRORS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
POSITION*	Integer
CHANGEDAT	Timestamp(3)
DETAILS	VarChar(256)

Column Name	Data Type
ERROREDAT	Timestamp(3)
EXTERNALOBJECTID	Integer
OBJECTTYPE	VarChar(64)
PORTALID	Integer
STATUS	VarChar(64)
STOREID	VarChar(64)
TYPE	VarChar(64)

EMAILSUBSCRIPTIONS

Endpoint

`https://api.hubapi.com/email/public/v1/subscriptions`

Columns

The EMAILSUBSCRIPTIONS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
ACTIVE	Boolean
CATEGORY	VarChar(64)
CHANNEL	VarChar(64)
DESCRIPTION	VarChar(88)
INTERNAL	Boolean
INTERNALNAME	VarChar(64)
NAME	VarChar(64)
PORTALID	Integer

ENGAGEMENTASSOCIATEDCOMPANIES

Endpoint

`https://api.hubapi.com/engagements/v1/engagements/paged`

Parent Table

ENGAGEMENTS

Columns

The ENGAGEMENTASSOCIATEDCOMPANIES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ENGAGEMENTS_ENGAGEMENT_ID*	BigInt	References: ENGAGEMENTS.ID
POSITION*	Integer	
ENGAGEMENTASSOCIATEDCOMPANY	BigInt	

ENGAGEMENTASSOCIATEDCONTACTIDS

Endpoint

`https://api.hubapi.com/engagements/v1/engagements/paged`

Parent Table

ENGAGEMENTS

Columns

The ENGAGEMENTASSOCIATEDCONTACTIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ENGAGEMENTS_ENGAGEMENT_ID*	BigInt	References: ENGAGEMENTS.ID
POSITION*	Integer	
ENGAGEMENTASSOCIATEDCONTACTID	BigInt	

ENGAGEMENTASSOCIATEDCONTENTIDS

Endpoint

`https://api.hubapi.com/engagements/v1/engagements/paged`

Parent Table

ENGAGEMENTS

Columns

The ENGAGEMENTASSOCIATEDCONTENTIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ENGAGEMENTS_ENGAGEMENT_ID*	BigInt	References: ENGAGEMENTS.ID
POSITION*	Integer	
ENGAGEMENTASSOCIATEDCONTENTID	BigInt	

ENGAGEMENTASSOCIATEDDEALIDS

Endpoint

`https://api.hubapi.com/engagements/v1/engagements/paged`

Parent Table

ENGAGEMENTS

Columns

The ENGAGEMENTASSOCIATEDDEALIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ENGAGEMENTS_ENGAGEMENT_ID*	BigInt	References: ENGAGEMENTS.ID
POSITION*	Integer	
ENGAGEMENTASSOCIATEDDEALID	BigInt	

ENGAGEMENTASSOCIATEDOWNERIDS

Endpoint

```
https://api.hubapi.com/engagements/v1/engagements/paged
```

Parent Table

ENGAGEMENTS

Columns

The ENGAGEMENTASSOCIATEDOWNERIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ENGAGEMENTS_ENGAGEMENT_ID*	BigInt	References: ENGAGEMENTS.ID
POSITION*	Integer	
ENGAGEMENTASSOCIATEDOWNERID	BigInt	

ENGAGEMENTASSOCIATEDQUOTEIDS

Endpoint

```
https://api.hubapi.com/engagements/v1/engagements/paged
```

Parent Table

ENGAGEMENTS

Columns

The ENGAGEMENTASSOCIATEDQUOTEIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ENGAGEMENTS_ENGAGEMENT_ID*	BigInt	References: ENGAGEMENTS.ID
POSITION*	Integer	
ENGAGEMENTASSOCIATEDQUOTEID	BigInt	

ENGAGEMENTASSOCIATEDTICKETIDS

Endpoint

`https://api.hubapi.com/engagements/v1/engagements/paged`

Parent Table

ENGAGEMENTS

Columns

The ENGAGEMENTASSOCIATEDTICKETIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ENGAGEMENTS_ENGAGEMENT_ID*	BigInt	References: ENGAGEMENTS.ID
POSITION*	Integer	
ENGAGEMENTASSOCIATEDTICKETID	BigInt	

ENGAGEMENTASSOCIATEDWORKFLOWIDS

Endpoint

`https://api.hubapi.com/engagements/v1/engagements/paged`

Parent Table

ENGAGEMENTS

Columns

The ENGAGEMENTASSOCIATEDWORKFLOWIDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ENGAGEMENTS_ENGAGEMENT_ID*	BigInt	References: ENGAGEMENTS.ID
POSITION*	Integer	
ENGAGEMENTASSOCIATEDWORKFLOWID	BigInt	

ENGAGEMENTS

Endpoint

`https://api.hubapi.com/engagements/v1/engagements/paged`

Columns

The ENGAGEMENTS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
ATTACHMENTS	JSON(16777215)
BODY	VarChar(64)
CREATEDAT	Timestamp(3)
CREATEDBY	Integer
DATETIME	Timestamp(3)
ENGAGEMENT_BODYPREVIEW	VarChar(64)
ISACTIVE	Boolean
MODIFIEDBY	Integer
OWNERID	Integer
PORTALID	Integer
STATUS	VarChar(64)
SUBJECT	VarChar(64)
TYPE	VarChar(64)
UPDATEDAT	Timestamp(3)

ENGAGEMENTSSCHEDULEDTASKS

Endpoint

`https://api.hubapi.com/engagements/v1/engagements/paged`

Parent Table

ENGAGEMENTS

Columns

The ENGAGEMENTSSCHEDULEDTASKS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
ENGAGEMENTS_ENGAGEMENT_ID*	BigInt	References: ENGAGEMENTS.ID
POSITION*	Integer	
DATETIME	Timestamp(3)	
ENGAGEMENTID	BigInt	
ENGAGEMENTTYPE	VarChar(64)	
PORTALID	Integer	
TASKTYPE	VarChar(64)	
UUID	VarChar(64)	

EVENTS

Endpoint

`https://api.hubapi.com/reports/v2/events`

Columns

The EVENTS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	VarChar(64)
LABEL	VarChar(64)
NAME	VarChar(64)
STATUS	VarChar(64)

FILES

Endpoint

`https://api.hubapi.com/filemanager/api/v2/files`

Columns

The FILES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
ALTKEY	VarChar(130)
ALTURL	VarChar(172)
CREATEDAT	Timestamp(3)
DELETEDAT	Timestamp(3)
EXTENSION	VarChar(64)
FILE_HASH	VarChar(64)
FOLDERID	BigInt
FRIENDLYURL	VarChar(174)
HEIGHT	Integer
ICONFRIENDLYURL	VarChar(187)
ICONIMAGENAME	VarChar(64)
ICONS3URL	VarChar(186)
ISARCHIVED	Boolean
MEDIUMFRIENDLYURL	VarChar(190)
MEDIUMIMAGENAME	VarChar(64)
MEDIUMS3URL	VarChar(189)
META_ALLOWS_ANONYMOUS_ACCESS	Boolean
META_URL_SCHEME	VarChar(64)
NAME	VarChar(64)

Column Name	Data Type
PORTAL_ID	Integer
S3_URL	VarChar(172)
SIZE	Integer
THUMBFRIENDLYURL	VarChar(189)
THUMBIMAGENAME	VarChar(64)
THUMBS3URL	VarChar(187)
TITLE	VarChar(64)
TYPE	VarChar(64)
UPDATEDAT	Timestamp(3)
URL	VarChar(174)
WIDTH	Integer

FOLDERS

Endpoint

```
https://api.hubapi.com/filemanager/api/v2/folders
```

Columns

The FOLDERS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
CATEGORY	Integer
CDNPURGEEMBARGOTIME	VarChar(64)
CREATEDAT	Timestamp(3)
DELETED	Boolean
DELETEDAT	Timestamp(3)
FULLPATH	VarChar(64)

Column Name	Data Type
HIDDEN	Boolean
NAME	VarChar(64)
PARENTFOLDERID	BigInt
PORTALID	Integer
UPDATEDAT	Timestamp(3)

FORMFIELDGROUPS

Endpoint

`https://api.hubapi.com/forms/v2/forms`

Parent Table

FORMS

Columns

The FORMFIELDGROUPS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
FORMS_PORTALID*	Integer	References: FORMS.PORTALID
POSITION*	Integer	
CONTENT	VarChar(64)	
DEFAULT_	Boolean	
ISSMARTGROUP	Boolean	
TYPE	VarChar(64)	

FORMFIELDOPTIONS

Endpoint

`https://api.hubapi.com/forms/v2/fields/{FormGUID}`

Parent Table

FORMFIELDS

Columns

The FORMFIELDOPTIONS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
FORMFIELDS_FORMGUID*	VarChar(64)	References: FORMFIELDS.FORMGUID
FORMFIELDS_LABEL*	VarChar(64)	References: FORMFIELDS.LABEL
POSITION*	Integer	
DISCRIPTION	VarChar(64)	
DISPLAYORDER	Integer	
DOUBLEDATA	Double	
ISHIDDEN	Boolean	
LABEL	VarChar(64)	
READONLY	Boolean	
VALUE	VarChar(64)	

FORMFIELDS

Endpoint

`https://api.hubapi.com/forms/v2/fields/{FormGUID}`

Columns

The FORMFIELDS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
FORMGUID*	VarChar(64)
LABEL*	VarChar(64)
DEFAULTVALUE	VarChar(64)
DISPLAYORDER	Integer

Column Name	Data Type
ENABLED	Boolean
FIELDTYPE	VarChar(64)
GROUPNAME	VarChar(64)
HIDDEN	Boolean
ISSMARTFIELD	Boolean
NAME	VarChar(75)
REQUIRED	Boolean
TYPE	VarChar(64)
VALIDATION_VALIDATIONBLOCKEDEMAILADDRESSES	JSON(16777215)
VALIDATIONDATA	VarChar(64)
VALIDATIONMESSAGE	VarChar(64)
VALIDATIONNAME	VarChar(64)
VALIDATIONUSEDEFAULTBLOCKLIST	Boolean

FORMFIELDSELECTEDOPTIONS

Endpoint

`https://api.hubapi.com/forms/v2/fields/{FormGUID}`

Parent Table

FORMFIELDS

Columns

The FORMFIELDSELECTEDOPTIONS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
FORMFIELDS_FORMGUID*	VarChar(64)	References: FORMFIELDS.FORMGUID
FORMFIELDS_LABEL*	VarChar(64)	References: FORMFIELDS.LABEL

Column Name	Data Type	Notes
POSITION*	Integer	
FORMFIELDSELECTEDOPTION	VarChar(64)	

FORMS

Endpoint

`https://api.hubapi.com/forms/v2/forms`

Columns

The FORMS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PORTALID*	Integer
ACTION	VarChar(64)
CAMPAIGNGUID	VarChar(64)
CREATEDAT	Timestamp(3)
CSSCLASS	VarChar(64)
DELETEDAT	Timestamp(3)
ENDATTIMESTAMP	Timestamp(3)
FOLLOWUPID	VarChar(64)
FORMTYPE	VarChar(64)
GUID	GUID
IGNORECURRENTVALUES	Boolean
INLINEMESSAGE	VarChar(64)
ISDELETABLE	Boolean
ISPUBLISHED	Boolean
LEADNURTURINGCAMPAIGNID	VarChar(64)
METADATA	JSON(16777215)

Column Name	Data Type
METHOD	VarChar(64)
MIGRATEDFROM	VarChar(64)
MULTIVARIATETEST_CONTROLID	VarChar(64)
MULTIVARIATETEST_FINISHED	Boolean
MULTIVARIATETEST_VARIANTS	JSON(16777215)
MULTIVARIATETEST_WINNINGVARIANTID	VarChar(64)
NAME	VarChar(88)
NOTIFYRECIPIENTS	VarChar(64)
PARENTID	Integer
PERFORMABLEHTML	VarChar(64)
PUBLISHAT	Timestamp(3)
PUBLISHEDAT	Timestamp(3)
REDIRECT	VarChar(64)
STARTATTIMESTAMP	Timestamp(3)
SUBMITTEXT	VarChar(64)
UNPUBLISHAT	Timestamp(3)
UPDATEDAT	Timestamp(3)

LINEITEMS

Endpoint

```
https://api.hubapi.com/crm-objects/v1/objects/line_items/paged
```

Columns

The LINEITEMS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
OBJECTTYPE*	VarChar(64)

Column Name	Data Type
DESCRIPTION	VarChar(64)
HSPRODUCTID	Integer
ISDELETED	Boolean
NAME	VarChar(64)
OBJECTID	Integer
PORTALID	Integer
PRICE	Integer
QUANTITY	Integer

LINEITEMS_V3

Endpoint

```
https://api.hubapi.com/crm/v3/objects/line_items
```

Columns

The LINEITEMS_V3 table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
ARCHIVED	Boolean
CREATEDAT	Timestamp(6)
DESCRIPTION	VarChar(64)
HSPRODUCTID	Integer
NAME	VarChar(64)
PRICE	Integer
QUANTITY	Integer
UPDATEDAT	Timestamp(6)

MARKETINGEMAILS

Endpoint

`https://api.hubapi.com/marketing-emails/v1/emails`

Columns

The MARKETINGEMAILS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
AB	Boolean
ABHOURSTOWAIT	Integer
ABSAMPLESIZEDEFAULT	VarChar(64)
ABSAMPLINGDEFAULT	VarChar(64)
ABSOLUTEURL	VarChar(120)
ABSUCCESSMETRIC	VarChar(64)
ABTESTPERCENTAGE	Integer
ABVARIATION	Boolean
ANALYTICSPAGEID	BigInt
ANALYTICSPAGETYPE	VarChar(64)
ARCHIVED	Boolean
AUTHOR	VarChar(64)
AUTHORAT	BigInt
AUTHORNAME	VarChar(64)
CAMPAIGN	GUID
CAMPAIGNNAME	VarChar(64)
CAMPAIGNUTM	VarChar(64)
CANSPAMSETTINGSID	BigInt

Column Name	Data Type
CATEGORYID	Integer
CONTENTTYPECATEGORY	Integer
CREATED	Timestamp(3)
CREATEDBYID	Integer
CREATEPAGE	Boolean
CURRENTLYPUBLISHED	Boolean
CURRENTSTATE	VarChar(64)
CUSTOMREPLYTO	VarChar(64)
CUSTOMREPLYTOENABLED	Boolean
DOMAIN	VarChar(64)
EMAILBODY	VarChar(130)
EMAILBODYPLAINTEXT	VarChar(64)
EMAILCAMPAIGNGROUPID	Integer
EMAILNOTE	VarChar(64)
EMAILTYPE	VarChar(64)
FEEDBACKSURVEYID	Integer
FREEZEDATE	BigInt
FROMNAME	VarChar(64)
HASCONTENTACCESSRULES	Boolean
HTMLTITLE	VarChar(64)
ISGRAYMAILSUPPRESSIONENABLED	Boolean
ISPUBLISHED	Boolean
LANGUAGE	VarChar(64)
LIVEDOMAIN	VarChar(64)
MAILINGLISTSEXCLUDED	JSON(16777215)
MAILINGLISTSINCLUDED	JSON(16777215)

Column Name	Data Type
MAXRSSENTRIES	Integer
METADESCRIPTION	VarChar(64)
NAME	VarChar(64)
PAGEREDIRECTED	Boolean
PORTALID	Integer
PREVIEWKEY	VarChar(64)
PROCESSINGSTATUS	VarChar(64)
PUBLISHDATE	Date
PUBLISHEDAT	Timestamp(3)
PUBLISHEDBYEMAIL	VarChar(64)
PUBLISHEDBYID	Integer
PUBLISHEDBYNAME	VarChar(64)
PUBLISHEDURL	VarChar(120)
PUBLISHIMMEDIATELY	Boolean
REPLYTO	VarChar(64)
RESOLVEDDOMAIN	VarChar(64)
RSSEMAILBYTEXT	VarChar(64)
RSSEMAILCLICKTHROUGHTEXT	VarChar(64)
RSSEMAILCOMMENTTEXT	VarChar(64)
RSSEMAILENTRYTEMPLATEENABLED	Boolean
RSSEMAILURL	VarChar(64)
SLUG	VarChar(78)
SUBCATEGORY	VarChar(64)
SUBJECT	VarChar(64)
SUBSCRIPTION	Integer
SUBSCRIPTIONNAME	VarChar(64)

Column Name	Data Type
TEAMPERMS	JSON(16777215)
TEMPLATEPATH	VarChar(87)
TRANSACTIONAL	Boolean
UNPUBLISHEDAT	Timestamp(3)
UPDATED	Timestamp(3)
UPDATEDBYID	Integer
URL	VarChar(120)
USERPERMS	JSON(16777215)
USERSSHEADLINEASSUBJECT	Boolean
VIDSEXCLUDED	JSON(16777215)
VIDSINCLUDED	JSON(16777215)
VISIBLETOALL	Boolean
WIDGETS	JSON(16777215)

OWNERS

Endpoint

`https://api.hubapi.com/owners/v2/owners/`

Columns

The OWNERS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PORTALID*	Integer
ACTIVESALESFORCEID	VarChar(64)
ACTIVEUSERID	Integer
CREATEDAT	Timestamp(3)
EMAIL	VarChar(76)

Column Name	Data Type
FIRSTNAME	VarChar(64)
HASCONTACTSACCESS	Boolean
ISACTIVE	Boolean
LASTNAME	VarChar(64)
OWNERID	Integer
TYPE	VarChar(9)
UPDATEDAT	Timestamp(3)
USERIDINCLUDINGINACTIVE	Integer

OWNERSREMOTELISTS

Endpoint

`https://api.hubapi.com/owners/v2/owners/`

Parent Table

OWNERS

Columns

The OWNERSREMOTELISTS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
OWNERS_PORTALID*	Integer	References: OWNERS.PORTALID
POSITION*	Integer	
ACTIVE	Boolean	
ID	Integer	
OWNERID	Integer	
PORTALID	Integer	
REMOTEID	Integer	
REMOTETYPE	VarChar(10)	

PAGES

Endpoint

```
https://api.hubapi.com/content/api/v2/pages
```

Columns

The PAGES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
CREATEDAT	Timestamp(3)
CURRENTLIVEDOMAIN	VarChar(64)
DELETEDAT	Integer
ISARCHIVED	Boolean
NAME	VarChar(79)
PUBLISHDATE	BigInt
SLUG	VarChar(64)
SUBCATEGORY	VarChar(64)
UPDATEDAT	BigInt
URL	VarChar(91)

PRODUCTS

Endpoint

```
https://api.hubapi.com/crm-objects/v1/objects/products/{ObjectId}
```

Columns

The PRODUCTS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PORTALID*	Integer

Column Name	Data Type
DESCRIPTION	VarChar(64)
ISDELETED	Boolean
NAME	VarChar(64)
OBJECTID	Integer
OBJECTTYPE	VarChar(64)
PRICE	Integer

PRODUCTS_V3

Endpoint

```
https://api.hubapi.com/crm/v3/objects/products
```

Columns

The PRODUCTS_V3 table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
ARCHIVED	Boolean
CREATEDAT	Timestamp(9)
PROPERTIES_DESCRIPTION	VarChar(64)
PROPERTIES_NAME	VarChar(64)
PROPERTIES_PRICE	Integer
UPDATEDAT	Timestamp(9)

QUOTES_V3

Endpoint

```
https://api.hubapi.com/crm/v3/objects/quotes
```

Columns

The QUOTES_V3 table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
ARCHIVED	Boolean
CREATEDAT	Timestamp(6)
PROPERTIES_HS_EXPIRATION_DATE	Timestamp(9)
PROPERTIES_HS_PUBLIC_URL_KEY	VarChar(64)
PROPERTIES_HS_STATUS	VarChar(64)
PROPERTIES_HS_TITLE	VarChar(64)
UPDATEDAT	Timestamp(6)

SOCIALMEDIACHANNELS

Endpoint

`https://api.hubapi.com/broadcast/v1/channels/setting/publish/current`

Columns

The SOCIALMEDIACHANNELS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PORTALID*	Integer
ACCOUNTGUID	GUID
ACCOUNTSLUG	VarChar(64)
ACCOUNTTYPE	VarChar(64)
CHANNELGUID	GUID
CHANNELID	Integer
CHANNELKEY	VarChar(64)
CHANNELSLUG	VarChar(64)

Column Name	Data Type
CREATEDAT	Timestamp(3)
ISACTIVE	Boolean
ISHIDDEN	Boolean
ISSHARED	Boolean
TYPE	VarChar(64)
UPDATEDAT	Timestamp(3)
USERNAME	VarChar(64)

SOCIALMEDIAMESSAGES

Endpoint

`https://api.hubapi.com/broadcast/v1/broadcasts`

Columns

The SOCIALMEDIAMESSAGES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
CONTENTBODY*	VarChar(256)
CAMPAIGNGUID	GUID
CAMPAIGNNAME	VarChar(64)
CHANNEL	VarChar(64)
CHANNELGUID	GUID
CLICKS	Integer
CLIENTTAG	VarChar(64)
CONTENTORIGINALBODY	VarChar(256)
CONTENTPHOTOURL	VarChar(512)
CONTENTUNCOMPRESSEDLINKS	VarChar(256)
CREATEDAT	Timestamp(3)

Column Name	Data Type
CREATEDBY	Integer
FINISHEDAT	Timestamp(3)
FOREIGNID	Integer
GROUPGUID	GUID
ISFAILED	Boolean
ISPENDING	Boolean
ISPUBLISHED	Boolean
ISRETRY	Boolean
MESSAGE	VarChar(256)
REMOTECONTENTID	VarChar(64)
REMOTECONTENTTYPE	VarChar(64)
STATUS	VarChar(64)
TRIGGERAT	VarChar(64)
UPDATEDBY	Integer

TASKS

Endpoint

`https://api.hubapi.com/calendar/v1/events/task`

Columns

The TASKS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
STARTDATE*	VarChar(64)
ENDDATE*	VarChar(64)
AVATARURL	VarChar(64)
CAMPAIGNGUID	GUID

Column Name	Data Type
CATEGORY	VarChar(64)
CATEGORYID	Integer
CONTENTGROUPID	Integer
CONTENTID	Integer
CREATEDBY	VarChar(64)
DESCRIPTION	VarChar(64)
EVENTDATE	BigInt
EVENTTYPE	VarChar(64)
ID	Integer
NAME	VarChar(64)
OWNERID	BigInt
PORTALID	Integer
PREVIEWKEY	VarChar(64)
RECURRING	Boolean
SOCIALDISPLAYNAME	VarChar(64)
SOCIALUSERNAME	VarChar(64)
STATE	VarChar(64)
TOPICIDS	JSON(16777215)
URL	VarChar(256)

TEMPLATES

Endpoint

```
https://api.hubapi.com/content/api/v2/templates
```

Columns

The TEMPLATES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
CATEGORYID	Integer
CDNMINIFIEDURL	VarChar(255)
CDNURL	VarChar(249)
DELETEDAT	Timestamp(3)
FOLDER	VarChar(84)
GENERATEDFROMLAYOUTID	BigInt
ISAVAILABLEFORNEWCONTENT	Boolean
ISFROMLAYOUT	Boolean
ISREADONLY	Boolean
LABEL	VarChar(60)
PATH	VarChar(109)
SOURCE	LongVarChar(95280)
UPDATEDAT	Timestamp(3)

TICKETPIPELINES

Endpoint

`https://api.hubapi.com/crm-pipelines/v1/pipelines/tickets`

Columns

The TICKETPIPELINES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
PIPELINEID*	Integer
ACTIVE	Boolean
CREATEDAT	Integer
DEFAULT_	Boolean

Column Name	Data Type
DISPLAYORDER	Integer
LABEL	VarChar(64)
OBJECTTYPE	VarChar(64)
OBJECTTYPEID	VarChar(64)
UPDATEDAT	VarChar(64)

TICKETPIPELINESTAGES

Endpoint

`https://api.hubapi.com/crm-pipelines/v1/pipelines/tickets`

Parent Table

TICKETPIPELINES

Columns

The TICKETPIPELINESTAGES table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type	Notes
TICKETPIPELINES_PIPELINEID*	Integer	References: TICKETPIPELINES.PIPELINEID
POSITION*	Integer	
ACTIVE	Boolean	
CREATEDAT	Integer	
DISPLAYORDER	Integer	
ISCLOSED	Boolean	
PROBABILITY	Double	
STAGEID	Integer	
STAGENAME	VarChar(64)	
UPDATEDAT	VarChar(64)	

TICKETS

Endpoint

```
https://api.hubapi.com/crm-objects/v1/objects/tickets/paged
```

Columns

The TICKETS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
OBJECTTYPE*	VarChar(64)
CONTENT	VarChar(64)
CREATEDBY	VarChar(64)
HSPIPELINE	Integer
HSPIPELINESTAGE	Integer
ISDELETED	Boolean
OBJECTID	Integer
PORTALID	Integer
SUBJECT	VarChar(64)

TICKETS_V3

Endpoint

```
https://api.hubapi.com/crm/v3/objects/tickets
```

Columns

The TICKETS_V3 table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	Integer
ARCHIVED	Boolean
CREATEDAT	Timestamp(9)

Column Name	Data Type
CREATEDBY	VarChar(64)
HSPIPELINE	Integer
HSPIPELINESTAGE	Integer
PROPERTIES_CONTENT	VarChar(64)
PROPERTIES_SUBJECT	VarChar(64)
UPDATEDAT	Timestamp(9)

URLMAPPINGS

Endpoint

`https://api.hubapi.com/url-mappings/v3/url-mappings`

Columns

The URLMAPPINGS table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
ID*	BigInt
CONTENTGROUPID	VarChar(64)
CREATEDAT	Timestamp(3)
DELETEDAT	Timestamp(3)
DESTINATION	VarChar(64)
ISMATCHFULLURL	Boolean
ISMATCHQUERYSTRING	Boolean
ISONLYAFTERNOTFOUND	Boolean
NAME	VarChar(64)
PORTALID	Integer
PRECEDENCE	Integer
REDIRECTSTYLE	Integer

Column Name	Data Type
ROUTEPREFIX	VarChar(256)
UPDATEDAT	Timestamp(3)

WORKFLOW

Endpoint

```
https://api.hubapi.com/automation/v3/workflows/{WorkflowId}
```

Columns

The WORKFLOW table contains the following columns. Columns marked with an asterisk comprise the primary key.

Column Name	Data Type
WORKFLOWID*	BigInt
ALLOWCONTACTTOTRIGGERMULTIPLETIMES	Boolean
ID	Integer
ISENABLED	Boolean
ISINTERNAL	Boolean
ISLISTENING	Boolean
NAME	VarChar(64)
ONLYEXECONBIZDAYS	Boolean
UPDATEDAT	Timestamp(3)