

APL64: Using SQL

Contents

Contents	1
Overview	2
Prerequisites:	3
Installation of SQL Server:	3
Display Summary Documentation:	5
 SQL Actions	5
BeginTransaction	5
Clear	5
Close	6
CommitTransaction	6
Count	8
Create	8
DateFormat	9
DBNull	10
DecimalSeparator	10
Delete	11
Exec	12
ExecDeleteQuery	13
ExecInsertQuery	14
ExecSelectQuery	17
ExecSelect&Get	19
ExecStoredProc	19
ExecStoredProcCmd	21
GetAllRecords	22
GetConnectionState	23
GetConnectionString	24
GetRecord	24
GetRecords	25

GroupSeparator.....	26
? or Help	27
Instances.....	27
InTransaction.....	28
New	28
Open.....	29
RegionalSettings.....	29
RollbackTransaction	29
Self.....	31
Learn Structured Query Language.....	32
Some Additional Examples	32
Create Table	32
Alter Table	33
Create/Alter Procedure.....	34
Update Table Record.....	36
Obtain Columns Information for a Table	38

Overview

The APL64 ☐SQL system function provides a structured query language (SQL)-based interface to a Microsoft SQL Server database.

The ☐SQL system function employs the [ADO.Net \(active data objects\) class for .Net](#) which has superior performance and features compared to previous technologies, such as ADODB, ODBC, MDAC, Jet, etc.

With the (no-cost) availability of [Microsoft SQL Server Express](#) database, scalable-to-enterprise-level database software is more affordable and the logical choice for professionally designed application systems.

The object model for ☐SQL includes methods and properties to easily create a secure, high-performance application system data interface.

Prerequisites:

To use ☐ SQL, the target workstation must have appropriate access to an existing SQL Server database. The SQL Server database software may be locally installed, installed on a local server or cloud based.

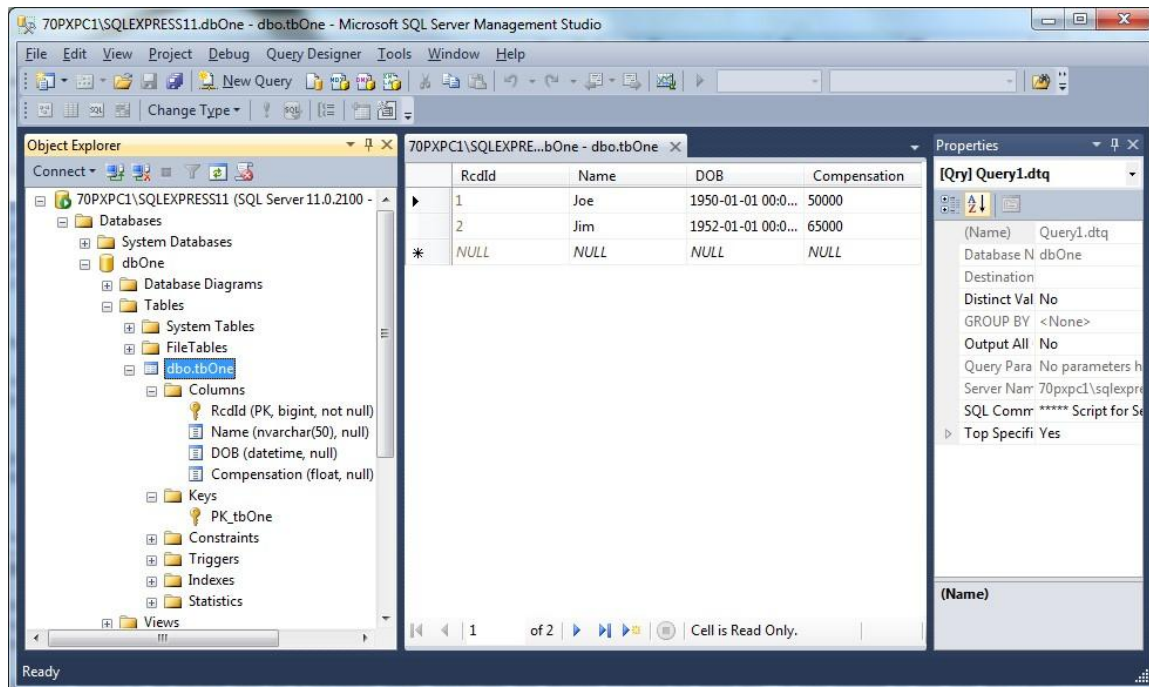
The examples in this document assume that a sample SQL Server database, named 'dbOne' is installed, running and accessible to the target workstation. The examples in this document also assume that this sample SQL Server database has a table, named 'tbOne', with sample columns and sample data.

The examples in this document are for demonstration purposes, so they may not incorporate enterprise-level security measures, may use a local workstation deployment that is not amenable to sharing data, and use a simplified [connection string](#). For simplicity, some examples do not include all APL64 executable statements which may be necessary to reproduce the illustrated output.

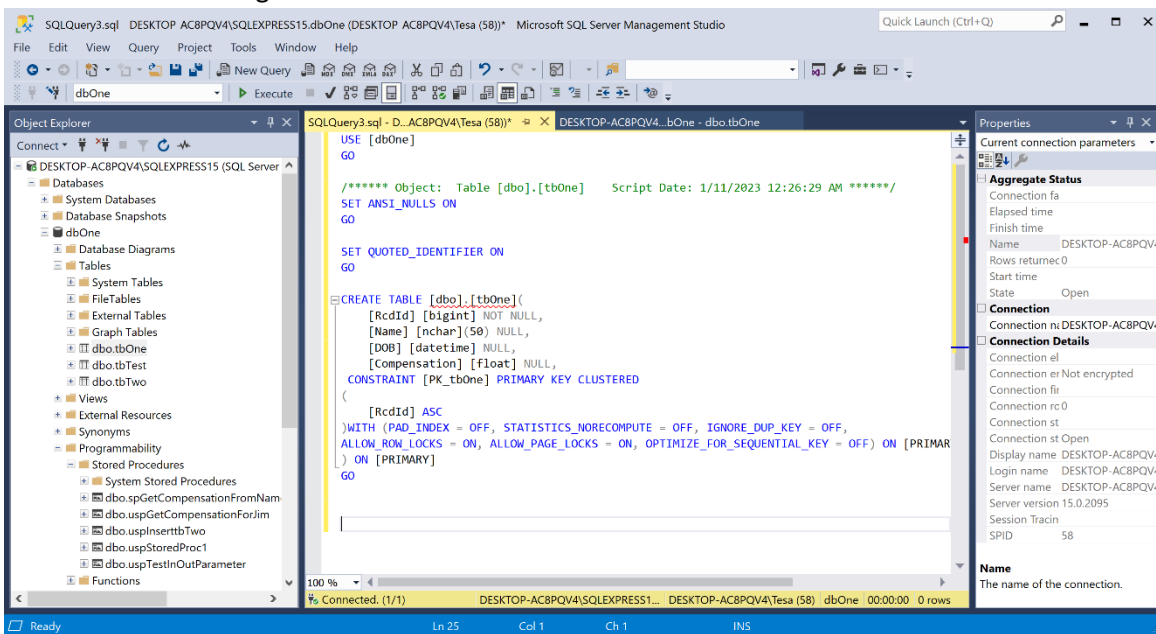
Installation of SQL Server:

Full installation details of an SQL Server database are beyond the scope of this document. Review the [Microsoft SQL Server Express](#) documentation for installation information or [Microsoft SQL Server trial version](#). Consider installing [Microsoft SQL Server Management Studio](#) as a tool to establish this instance.

The examples in the remainder of this document are based on the sample database, tables, and values illustrated in the above two documents. The initial state of the sample database as displayed in the SQL Server Management Studio:



The database name is 'dbOne' and the table in the database is 'tbOne' which can be created using the SQL Server Management Studio:



Insert the tbOne table sample data:

```
insert into tbOne values (1,'Joe','1960-01-01',50000);
insert into tbOne values (2,'Jim','1961-01-01',60000);
```

Create the sample stored procedure:

```
CREATE PROCEDURE [dbo].[spGetCompensationFromName]
-- Add the parameters for the stored procedure here
    @NameMatch nvarchar(45),
    @COMP float OUTPUT
```

```

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT @COMP = Compensation from tbOne WHERE Name =
@NameMatch
    Select @COMP
END

```

Display Summary Documentation:

SQL Actions

SQL actions operate on either the `SQL` object or an `SQL` instance. The `sql` object is a container for all `sql` instances in an APL64 instance. An `sql` instance is associated with a specific `Sql` database accessible to the target workstation user.

The left argument of the APL64 `sql` system function determines if the `sql` action applies to the `sql` object ('#' left argument) or `sql` instance (name of `sql` instance).

The name of an `sql` instance can be specified by:

- An APL64 text expression, e.g. 'myinstance', «myinstance»
- An APL64 variable with value equal to an `sql` instance name
- Elided, if the `SqlSelf` value is an `sql` instance name

`sql` action names are not case sensitive. `sql` instance names are case sensitive

BeginTransaction

This action applies to an `sql` instance. The `BeginTransaction` action, for the `Sql` instance named in the left argument, indicates that subsequent `sql` actions will be incorporated into an SQL transaction. An SQL transaction may contain multiple `sql` actions. An SQL transaction can either be committed or rolled back. The `BeginTransaction` action has no result.

[SQL transactions](#) are used to combine multiple SQL statements, possibly affecting multiple tables in an SQL database, so that they are either all performed or none are performed.

```

S←'#' SQL 'Create' 'S'
S SQL 'BeginTransaction'

```

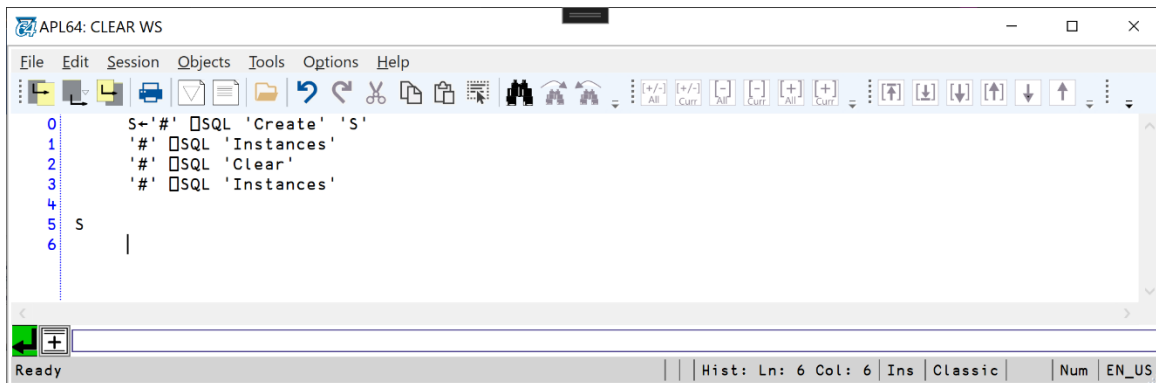
Clear

This action is performed on the `SQL` object. The `Clear` action closes the `Sql` database connection associated with any `SQL` instance and deletes all `SQL` instances in the current APL64 instance. The `Clear` action has no result.

```

S←'#' □SQL 'Create' 'S'
'#' □SQL 'Instances'
'#' □SQL 'Clear'
'#' □SQL 'Instances'

```



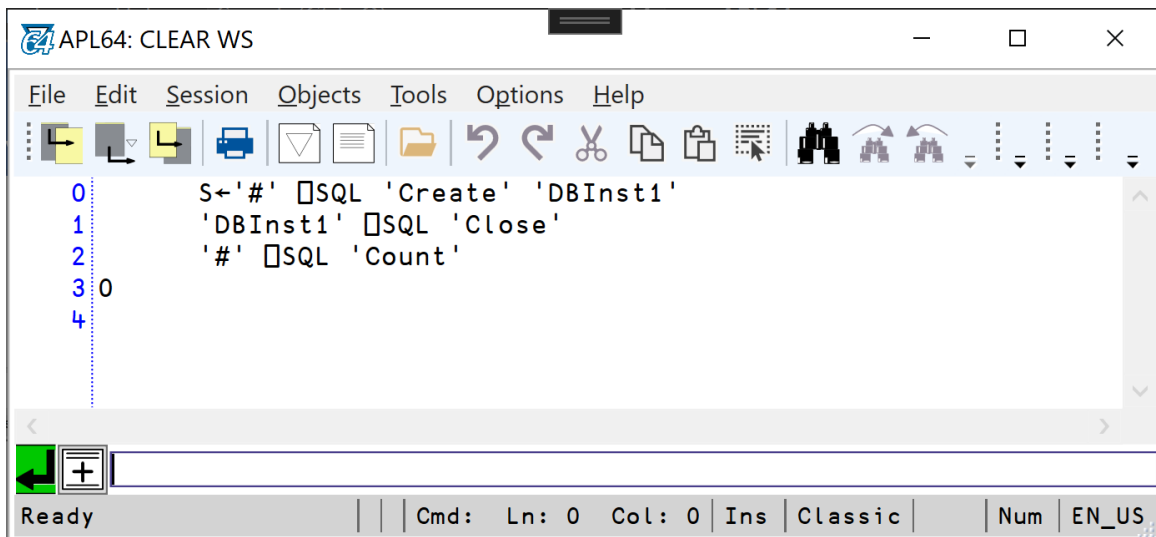
Close

The close action is performed on an ☐ SQL instance. The close action will close an existing connection and delete the associated ☐ SQL instance.

```

S←'#' □SQL 'Create' 'DBInst1'
'DBInst1' □SQL 'Close'
'#' □SQL 'Count'

```



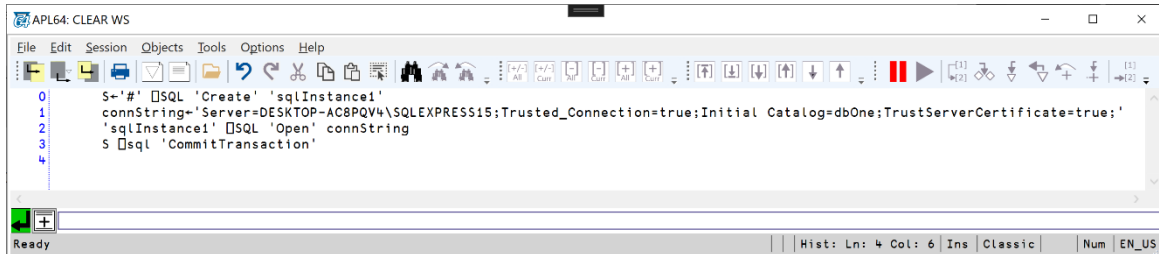
CommitTransaction

This action applies to an ☐ SQL instance. The CommitTransaction action causes the effects of the current transaction to be applied to the connected Microsoft Sql Server database. The CommitTransaction action has no effect if the BeginTransaction action has not occurred or if the RollbackTransaction action has occurred. Prior to committal or rollback of a transaction, the

effects of the existing SQL statements in the transaction are pending effects in the Sql database. The CommitTransaction action has no result.

```
S←'#' SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' SQL 'Open' connString
S SQL 'CommitTransaction'
```



```
S←'#' SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' SQL 'Open' connString
RecordSetId←S SQL 'ExecSelectQuery' 'Select Name from tbone'
S SQL 'GetAllRecords' RecordSetId
S SQL 'BeginTransaction'
S SQL 'ExecInsertQuery' 'tbone' 'RcdId, Name,DOB,Compensation' (3, 'Salmon' '1966-01-01'
90500.56)
RecordSetId2←S SQL 'ExecSelectQuery' 'Select Name from tbone;'
S SQL 'GetAllRecords' RecordSetId2
S SQL 'CommitTransaction'
S SQL 'ExecSelectQuery' 'Select Name from tbone;'
S SQL 'GetAllRecords' 104
S SQL 'ExecDeleteQuery' 'Delete From tbone Where RcdId=3;'
```

Note that the records in the RecordSetId2 data set include the pending effects of the ExecInsertQuery action and the records in the #104 data set include the effects of the committed ExecInsertQuery action:

```

0 S←'#' □SQL 'Create' 'sqlInstance1'
1 connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
2 'sqlInstance1' □SQL 'Open' connString
3 RecordSetId←S □SQL 'ExecSelectQuery' 'Select Name from tbone'
4 S □SQL 'GetAllRecords' RecordSetId
5 S □SQL 'BeginTransaction'
6 S □SQL 'ExecInsertQuery' 'tbone' 'RcdId, Name,DOB,Compensation' (3, 'Salmon' '1966-01-01' 90500.56)
7 RecordSetId2←S □SQL 'ExecSelectQuery' 'Select Name from tbone;'
8 S □SQL 'GetAllRecords' RecordSetId2
9 S □SQL 'CommitTransaction'
10 S □SQL 'ExecSelectQuery' 'Select Name from tbone;'
11 S □SQL 'GetAllRecords' 104
12 S □SQL 'ExecDeleteQuery' 'Delete From tbone Where RcdId=3;'
13 Joe
14 Jim
15 Joe
16 Jim
17 Salmon
18 104
19 Joe
20 Jim
21 Salmon
22 1
23

```

Count

This action is performed on the ☐SQL object. The Create action will create an ☐SQL instance with a user-provided name in the right argument. Multiple ☐SQL instances are possible in the same APL64 instance, so that multiple Sql Server databases may be conveniently accessed.

```

S←'#' □SQL 'Create' 'First'
T←'#' □SQL 'Create' 'Second'
□dr □←'#' □SQL 'Count'

```

```

0 S←'#' □SQL 'Create' 'First'
1 T←'#' □SQL 'Create' 'Second'
2 □dr □←'#' □SQL 'Count'
3 2
4 323
5

```

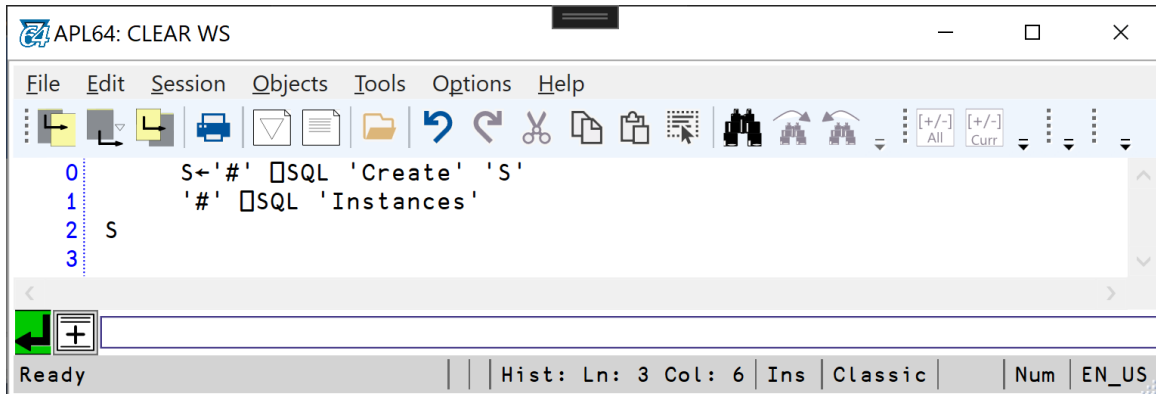
Create

This action is performed on the ☐SQL object. The Create action will create an ☐SQL instance with a user-provided name in the right argument. Multiple ☐SQL instances are possible in the same APL64 instance, so that multiple Sql Server databases may be conveniently accessed.

The Create action will not fail if the named instance already exists. The Create action will close and delete a pre-existing instance with the same name. The Create action for a particular Sql

Server database is generally used once in an APL64 instance. The Create action does not open a connection to an Sql Server database, use the Open action for that purpose. The result of a successful 'Create' action is a text vector containing the ☐SQL instance name.

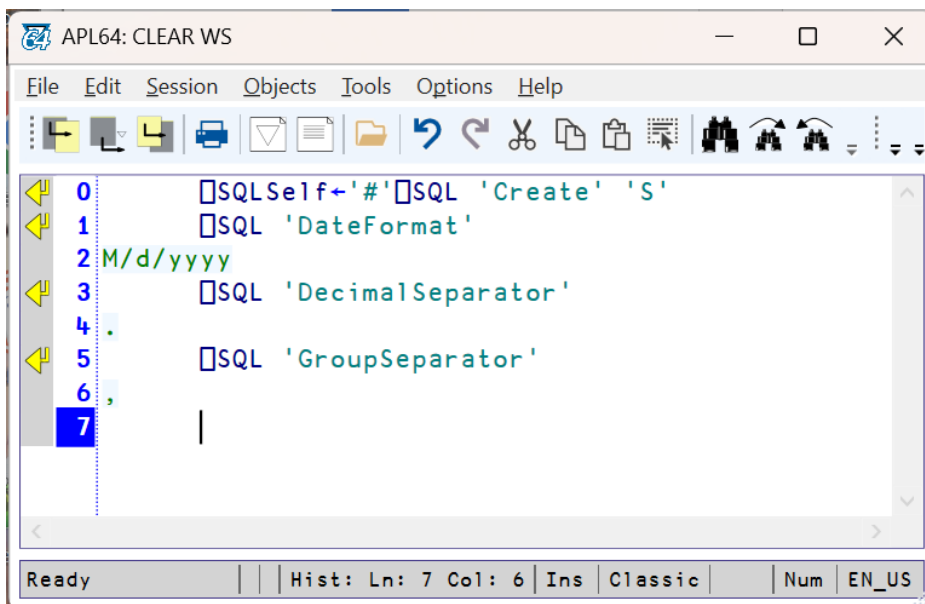
```
S←'#' ☐SQL 'Create' 'S'  
'#' ☐SQL 'Instances'
```



DateFormat

This action returns the current [.Net Short Date Pattern](#) text applicable to the specified ☐SQL instance.

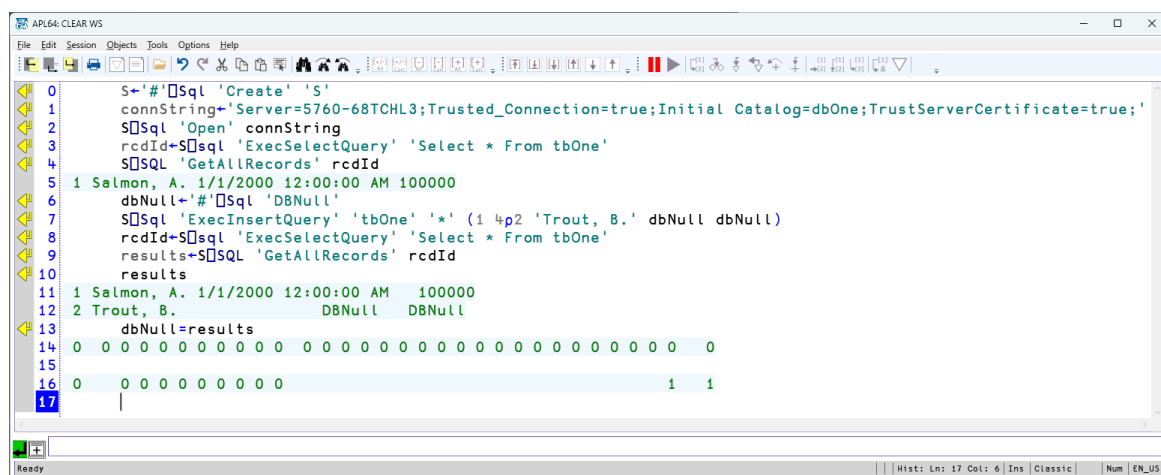
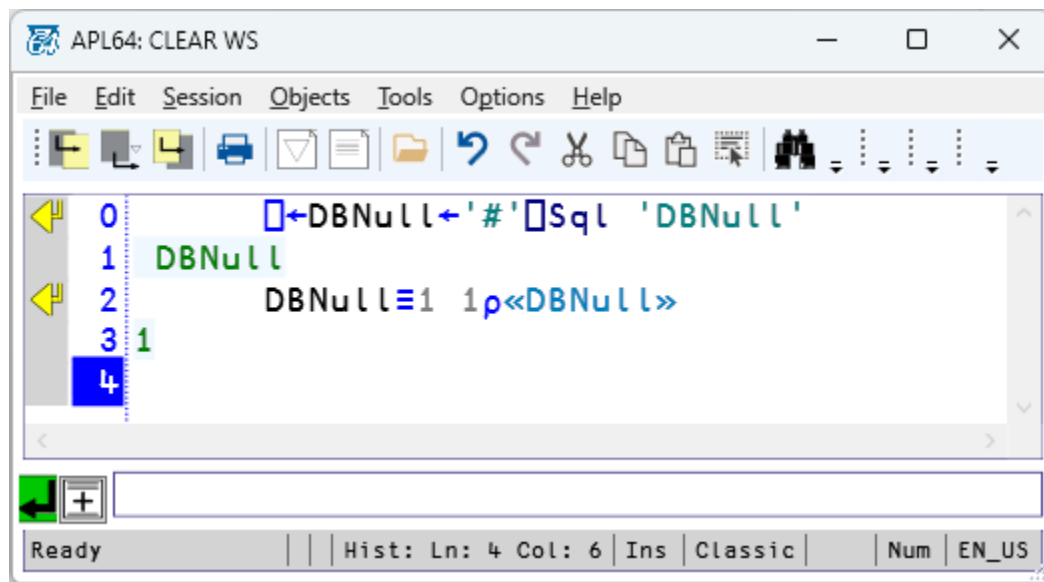
```
☐SQLSelf←'#' ☐SQL 'Create' 'S'  
☐SQL 'DateFormat'  
☐SQL 'DecimalSeparator'  
☐SQL 'GroupSeparator'
```



DBNull

This action returns a value which may be used to specify null values for the ExecInsertQuery and ExecStoredProc ☐ SQL instance actions. This value will also be used as the APL64 value associated with a DBNull value returned from a GetRecord, GetAllRecords, or ExecStoredProc action.

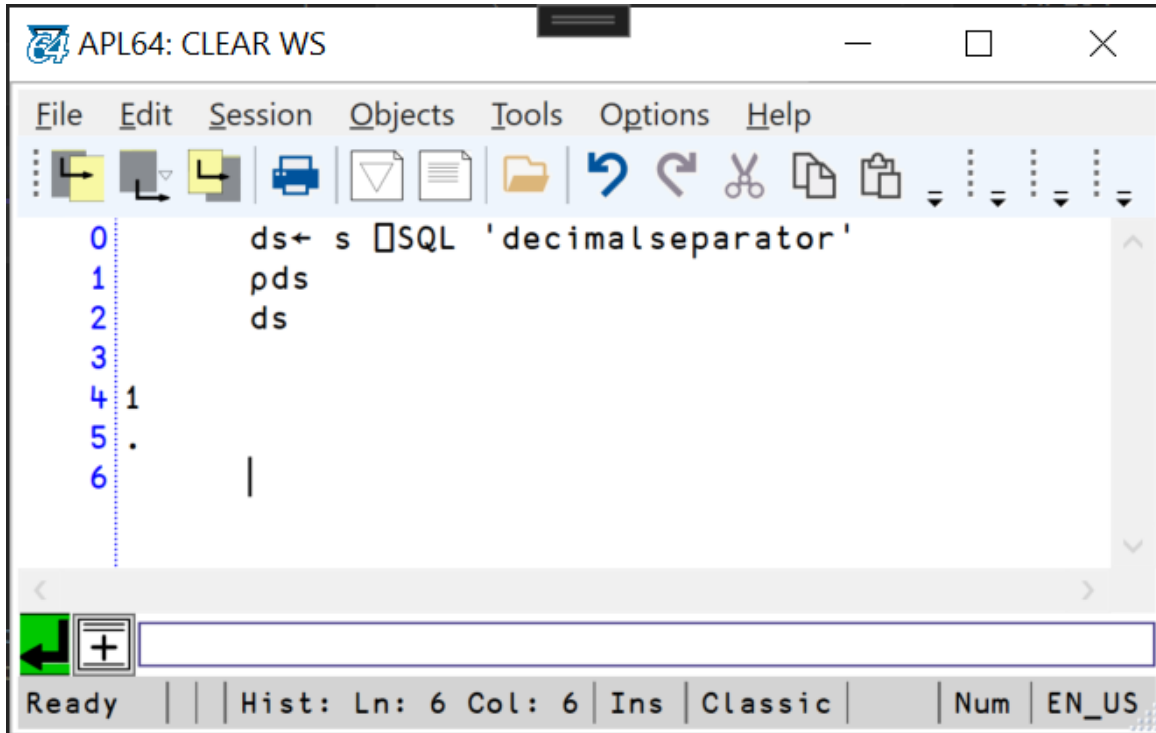
```
☐←DBNull←'#'☐Sql 'DBNull'  
DBNull≡1 1p«DBNull»
```



DecimalSeparator

This action returns the current [.Net Number Decimal Separator](#) text applicable to the specified ☐ SQL instance.

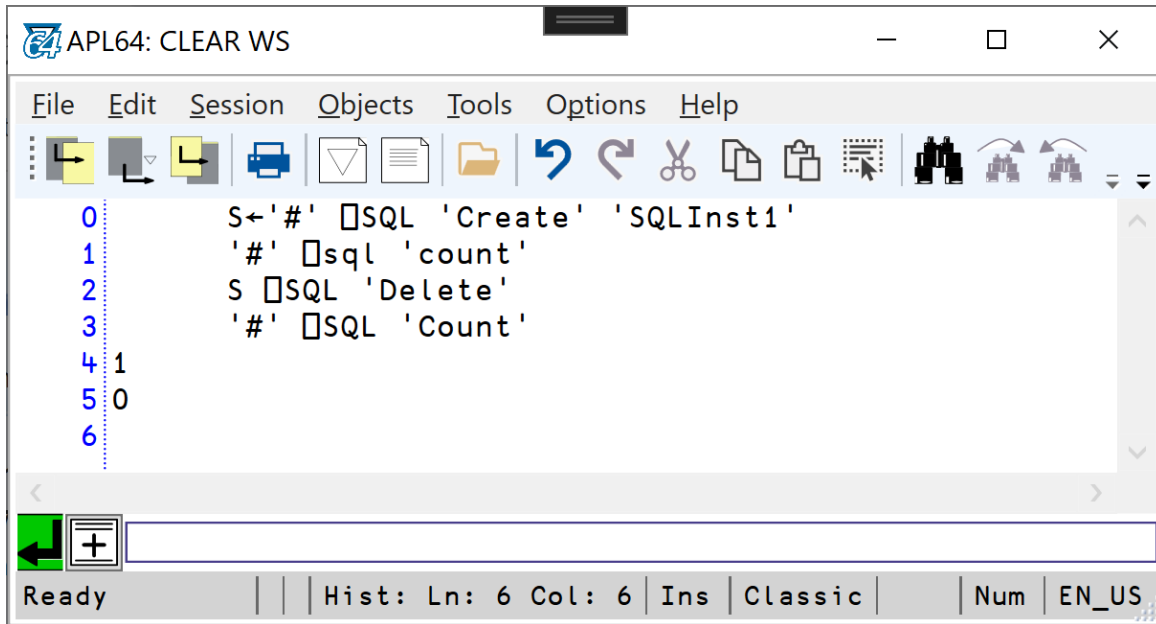
```
s←'#' □SQL 'Create' 'sInstanceName'
ds← s □SQL 'decimalseparator'
pds
ds
```



Delete

The Delete action is performed on an ☐SQL instance. The close action will close an existing connection and delete the associated ☐SQL instance.

```
S←'#' □SQL 'Create' 'SQLInst1'
'#' □sql 'count'
S □SQL 'Delete'
'#' □SQL 'Count'
```



Exec

This action applies to an ☐SQL instance. The ☐SQL Exec action returns the number of database rows, if any, affected by the executed SQL statement provided by the right argument. The ☐SQL Exec action cannot be used to return results of a Select SQL statement.

Syntax: [sqlInstance] ☐SQL 'Exec' cmd [subValue1] [subValue2] ...

APL64 values may be substituted into the executable expression. The substitution location in the executable expression is indicated by {n}, where n is the index, origin zero, of substitution value. Substitution values may be used more than once. All substitutions are performed prior to execution of the expression.

Example: ☐SQL 'Exec' 'Delete From tbOne Where RcdId={0};' 3

Example:

```

S←'#' □SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' □SQL 'Open' connString
'sqlInstance1' □SQL 'Exec' "Insert Into tbOne (RcdId,Name,DOB,Compensation) VALUES
(3,'Salmon','1966-01-01',90500.56);"
S □sql 'ExecSelectQuery' 'Select RcdId, Name, DOB, Compensation From tbOne'
S □SQL 'GetAllRecords' 102
S □SQL 'Exec' 'Delete From tbOne Where RcdId=3;'

```

```

0 S←'#' □SQL 'Create' 'sqlInstance1'
1 connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial C:
2 'sqlInstance1' □SQL 'Open' connString
3 'sqlInstance1' □SQL 'Exec' "Insert Into tbOne (RcdId,Name,DOB,Compensation) VALUE:
4 S □sql 'ExecSelectQuery' 'Select RcdId, Name, DOB, Compensation From tbOne'
5 S □SQL 'GetAllRecords' 102
6 S □SQL 'Exec' 'Delete From tbOne Where RcdId=3;'
7 1
8 102
9 1 Joe 1/1/1950 12:00:00 AM 50000.00
10 2 Jim 10/30/1952 8:15:34 AM 65000.00
11 3 Salmon 1/1/1966 12:00:00 AM 90500.56
12 1
13

```

Ready | Cmd: Ln: 0 Col: 0 | Ins | Classic | Num | EN_US

ExecDeleteQuery

Synonym: ExecDelete

Syntax: [sqlInstance] □SQL 'ExecDelete' cmd [subValue1] [subValue2] ...

This action applies to an □SQL instance. The right argument is the text of the SQL Delete statement. The result of the □SQL ExecDeleteQuery is the number of records affected, if any.

APL64 values may be substituted into the executable expression. The substitution location in the executable expression is indicated by {n}, where n is the index, origin zero, of substitution value. Substitution values may be used more than once. All substitutions are performed prior to execution of the expression.

Example: □SQL 'ExecDelete' 'Delete From tbOne Where RcdId={0};' 3

Example:

```

S←'#' □SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' □SQL 'Open' connString
'sqlInstance1' □SQL 'Exec' "Insert Into tbOne (RcdId,Name,DOB,Compensation) VALUES
(3,'Salmon','1966-01-01',90500.56);"
S □sql 'ExecSelectQuery' 'Select RcdId, Name, DOB, Compensation From tbOne'
S □SQL 'GetAllRecords' 102
S □SQL 'ExecDeleteQuery' 'Delete From tbOne Where RcdId=3;'

```

```
S □ sql 'ExecSelectQuery' 'Select RcdId, Name, DOB, Compensation From tbOne'
S □ SQL 'GetAllRecords' 103
```

```

0 S←'#' □SQL 'Create' 'sqlInstance1'
1 connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial C
2 'sqlInstance1' □SQL 'Open' connString
3 'sqlInstance1' □SQL 'Exec' "Insert Into tbOne (RcdId,Name,DOB,Compensation) VALUE:
4 S □sql 'ExecSelectQuery' 'Select RcdId, Name, DOB, Compensation From tbOne'
5 S □SQL 'GetAllRecords' 102
6 S □SQL 'ExecDeleteQuery' 'Delete From tbOne Where RcdId=3;'
7 S □sql 'ExecSelectQuery' 'Select RcdId, Name, DOB, Compensation From tbOne'
8 S □SQL 'GetAllRecords' 103
9 1
10 102
11 1 Joe 1/1/1950 12:00:00 AM 50000.00
12 2 Jim 10/30/1952 8:15:34 AM 65000.00
13 3 Salmon 1/1/1966 12:00:00 AM 90500.56
14 1
15 103
16 1 Joe 1/1/1950 12:00:00 AM 50000
17 2 Jim 10/30/1952 8:15:34 AM 65000
18 |

```

ExecInsertQuery

Synonym: ExecInsert

This action applies to an ☐ SQL instance.

Right arguments:

1. Sql table name
2. Sql column names: Comma delimited text with selected Sql table column names, or '*' to indicate all Sql table column names.
3. APL64 data: A matrix containing the APL64 values to be inserted. One row for each record to be inserted.

The order of the Sql column names and APL64 data matrix columns must be conformable.

Example:

```

S←'#' □SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' □SQL 'Open' connString

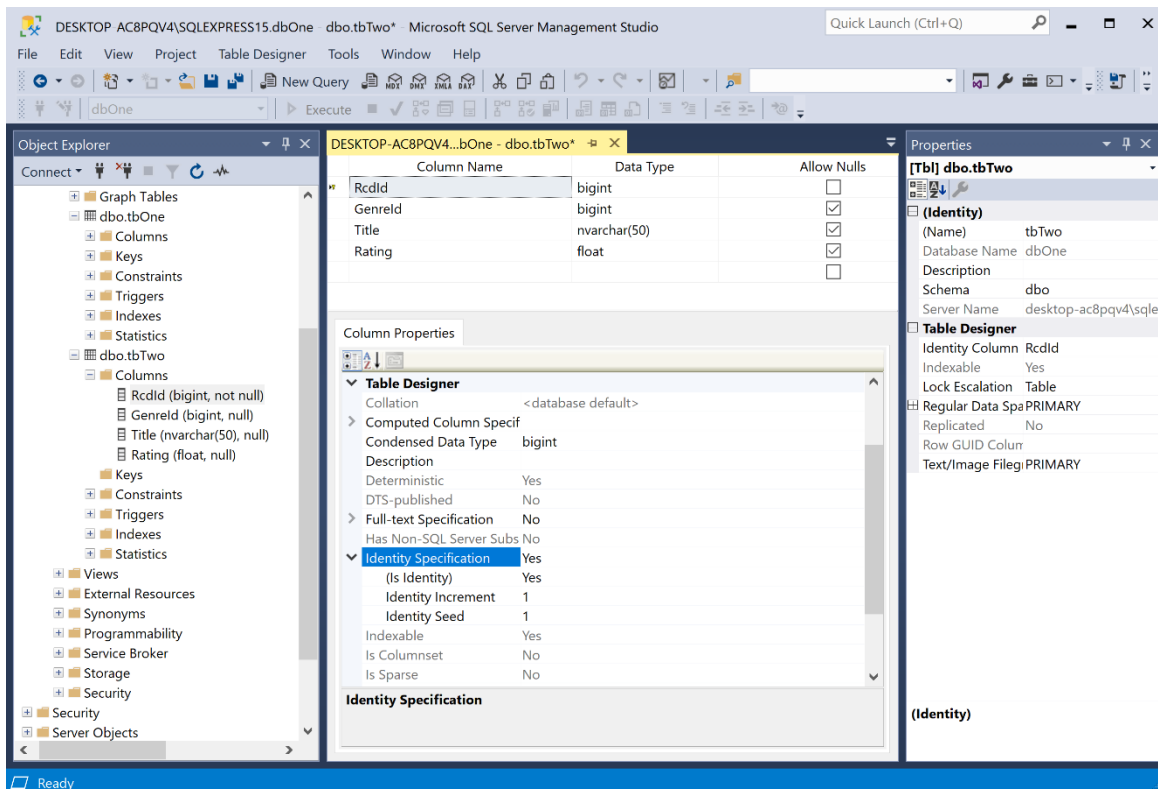
```

```
'sqlInstance1' SQL 'ExecInsertQuery' 'tbOne' 'RcdId, Name, DOB, Compensation' (1 4 p 3
'Mary' '04/23/1975' 5723.87)
```



The argument syntax for this method is:

- Argument #1: table name: The name of the database table into which records will be inserted
- Argument #2: fields: Either a string consisting of an asterisk (signifying all columns) or a comma-separated string of table column names for which values are provided in the records to be inserted.
- Argument #3: data: An array of APL64 data for the selected fields with one row for each record to be inserted and columns corresponding to each of the fields specified in Argument #2. If Argument #2 is '*' then Argument #3 must contain a value for every column in table Argument #1 that is not auto-generated.



The examples demonstrate using the ExecInsertQuery for a table without an Identity-type Primary Key field (tbOne) and a table with an Identity-type Primary Key field (tbTwo). The examples also demonstrate executing an insert statement specifying fields and an asterisk which indicates all fields.

'tbOne' does not contain such a field (Identity Column), so an Insert SQL statement should use the Exec or ExecuteSelectQuery methods. In the example below an additional table, 'tbTwo' was defined in the database which has an Identity field: RcdId.

```
S←'#' SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' SQL 'Open' connString
'sqlInstance1' SQL 'ExecInsertQuery' 'tbOne' 'RcdId, Name, DOB, Compensation' (1 4 p 3
'Mary' '04/23/1975' 5723.87)

S SQL 'ExecInsertQuery' 'tbTwo' 'GenreId, Title, Rating' (4 3 p 4 'Psycho' 7.87 4 'The Shining'
8.2 3 'Some Like It Hot' 9.4 3 'Annie Hall' 9.7)
```

```

S- '#' SQL 'Create' 'sqlInstance1'
connString='Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
'sqlInstance1' SQL 'Open' connString
'sqlInstance1' SQL 'ExecInsertQuery' 'tbOne' 'RcdId, Name, DOB, Compensation' (1 4 p 3 'Mary' '04/23/1975' 5723.87)
S SQL 'ExecInsertQuery' 'tbTwo' 'GenreId, Title, Rating' (4 3 p 4 'Psycho' 7.87 4 'The Shining' 8.2 3 'Some Like It Hot' 9.4 3 'Annie Hall' 9.7)

```

Using Microsoft SQL Server Management Studio to query the contents of both tables tbOne and tbTwo displays the result of the ExecInsertQuery statements.

```

S- '#' SQL 'Create' 'sqlInstance1'
connString='Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
'sqlInstance1' SQL 'Open' connString
'sqlInstance1' SQL 'ExecInsertQuery' 'tbOne' '*' (11 'Star2' '1/1/1999' 65450.63)
S SQL 'ExecSelectQuery' 'select @@Rowcount'
S SQL 'GetRecord' 102

```

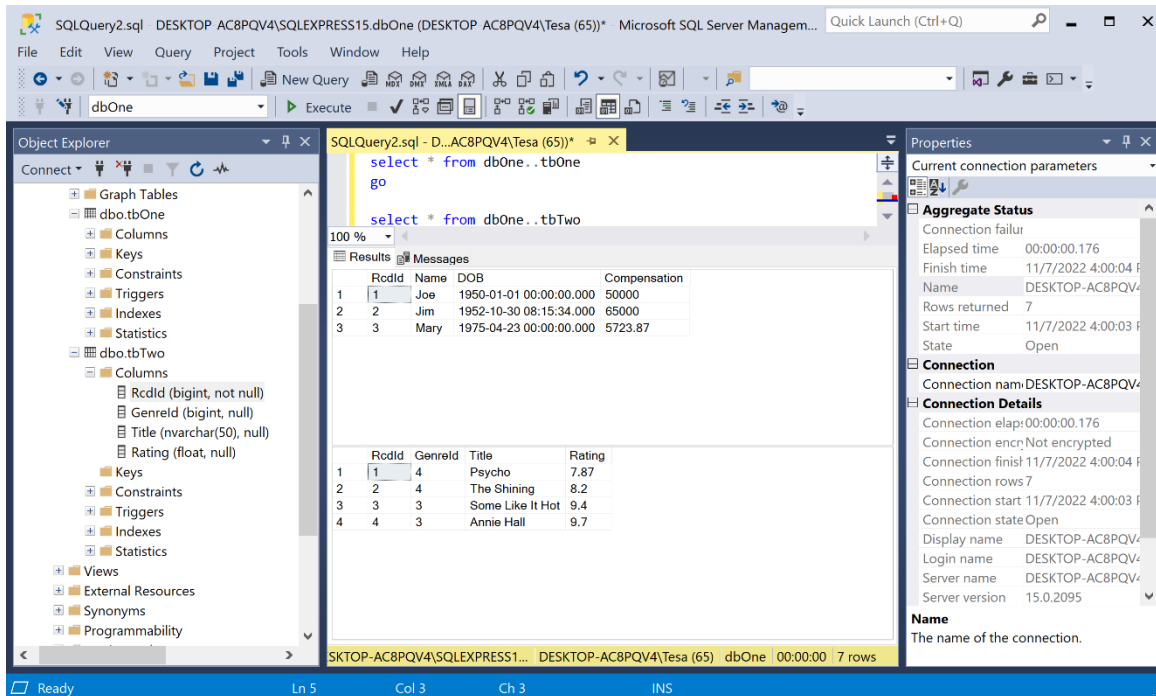
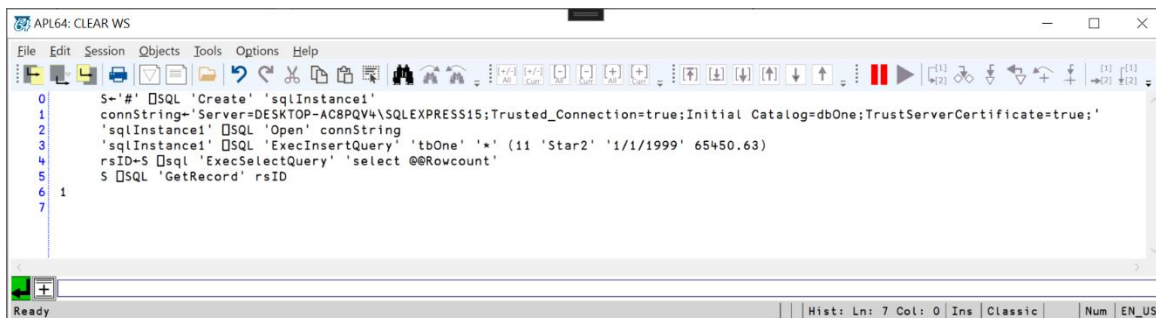


Illustration of insert query with no fields specified ('*'):

```
S←'#' SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' SQL 'Open' connString
'sqlInstance1' SQL 'ExecInsertQuery' 'tbOne' '*' (11 'Star2' '1/1/1999' 65450.63)
rsID←S SQL 'ExecSelectQuery' 'select @@Rowcount'
S SQL 'GetRecord' rsID
```



ExecSelectQuery

Synonym: ExecSelect

Syntax: matrix ← [sqlInstance] □SQL 'ExecSelect' cmd [subValue1] [subValue2] ...

The ExecSelectQuery action applies to an □sql instance. The action argument provides the SQL Select statement. Depending on the Sql data type of a column selected, it may be necessary to cast that data column values to a data type which has a representation in APL64.

The ExecSelectQuery does not return the result of the SQL Select statement. The result is an integer indicating the record set containing the result of the SQL Select statement. Use the 'GetRecord' or 'GetAllRecords' actions to access the specified record set.

The SQL-format query text argument to the □SQL ExecSelectQuery method, should yield only one record set. A query of the form 'query1;...;queryN', is not supported in APL64 and must be separated into individual queries. As an alternative to separate queries, a stored procedure can be created and executed using the □SQL ExecStoredProc method.

APL64 values may be substituted into the executable expression. The substitution location in the executable expression is indicated by {n}, where n is the index, origin zero, of substitution value. Substitution values may be used more than once. All substitutions are performed prior to execution of the expression.

Example: □SQL 'ExecSelect' 'Select * From tbOne Where RcdId={0};' 3

Example:

```
S←'#' □SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'
'sqlInstance1' □SQL 'Open' connString
S □sql 'ExecSelectQuery' 'Select RcdId, Name, DOB, Compensation From tbOne'
S □SQL 'GetAllRecords' 102
```

The screenshot shows the APL64: CLEAR WS application window. The code from the example is entered in the editor, and the output is displayed in a table below the code. The table has two columns: an index column (0-7) and a data column. The data for index 1 is 'Joe' and for index 2 is 'Jim'. The table also shows the date and time of execution (1/1/1950 12:00:00 AM and 10/30/1952 8:15:34 AM) and the compensation values (50000 and 65000).

Index	Data
0	S←'#' □SQL 'Create' 'sqlInstance1'
1	connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial C
2	'sqlInstance1' □SQL 'Open' connString
3	S □sql 'ExecSelectQuery' 'Select RcdId, Name, DOB, Compensation From tbOne'
4	S □SQL 'GetAllRecords' 102
5	1 Joe 1/1/1950 12:00:00 AM 50000
6	2 Jim 10/30/1952 8:15:34 AM 65000
7	

ExecSelect&Get

The ExecSelect&GetQuery action applies to an `SQL` instance. The action argument provides the SQL Select statement. Depending on the Sql data type of a column selected, it may be necessary to cast that data column values to a data type which has a representation in APL64.

Syntax: `matrix ← [sqlInstance] SQL 'ExecSelect&Get' cmd [subValue1] [subValue2] ...`

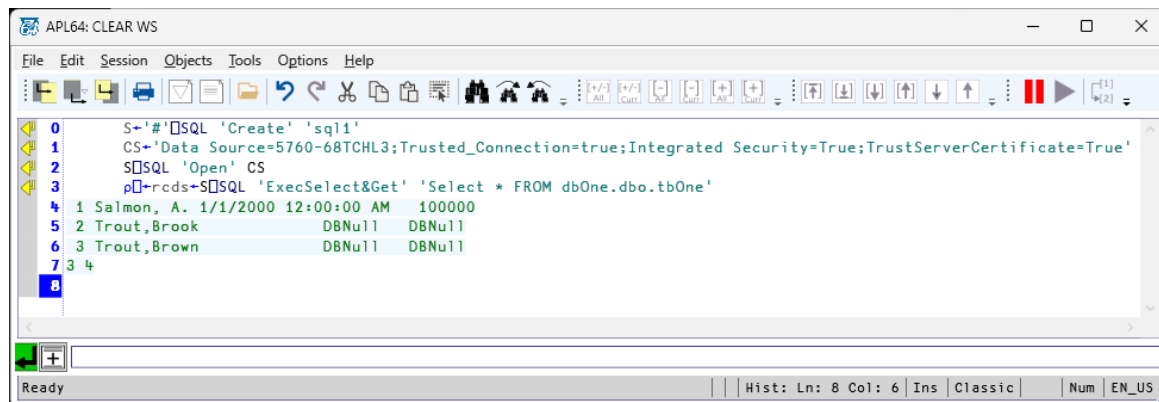
The ExecSelect&Get action combines the ExecSelect and GetAllRecords action, so that a matrix of values containing one row for each record in the data set is returned. Refer to the documentation of these two actions for details.

APL64 values may be substituted into the executable expression. The substitution location in the executable expression is indicated by `{n}`, where `n` is the index, origin zero, of substitution value. Substitution values may be used more than once. All substitutions are performed prior to execution of the expression.

Example: `SQL 'ExecSelect&Get' 'Select * From tbOne Where RcdId={0};' 3`

Example:

```
S←'#'SQL 'Create' 'sql1'
CS←'Data Source=5760-68TCHL3;Trusted_Connection=true;Integrated
Security=True;TrustServerCertificate=True'
SSQL 'Open' CS
p←rcds←SSQL 'ExecSelect&Get' 'Select * FROM dbOne.dbo.tbOne'
```



ExecStoredProc

This action can be used to run a stored procedure which has input, output and combined input/output parameters.

The action arguments are:

- An APL text vector (cmdText) containing the stored procedure name
- An APL comma-delimited text vector containing the parameter specifications. A parameter specification contains up to three text elements, each separated by a space:

- Case-sensitive parameter name (Required)
 - 'In', 'Out' or 'InOut' (Required)
 - APL64-compatible data type [Only for 'Out' parameters]
- An APL rank-1 vector of 'In' and 'InOut' parameter values. The order of the parameter values and the order of the 'In' and 'InOut' parameter names must be conformable.

Not all Sql data types can be represented in APL64, so the data types of stored procedure parameters must be carefully considered. For Sql data types which have no representation in APL64, the stored procedure must cast the parameter values to APL64-compatible data types.

- The Sql data types of In, InOut and Out stored procedure parameters must be compatible with APL64 data types.
- For Out parameters, there are two options to indicate the data type specified in the stored procedure definition:
 - Specify an APL64-compatible data type in the optional third element of the parameter specifications, e.g. 'paramName Out dataType'. If the provided data type text cannot be parsed into a DbType, the String type is used.
 - If the optional third element of the parameter specifications is not provided, e.g. 'paramName Out', the system will attempt to obtain the Out parameter's data type from the database schema of the stored procedure. If this attempt fails, the String data type is used.
 - A stored procedure Out parameter should have no 'place holder' in the dataVector.
- The APL64-compatible data types are Boolean, Double, Int32, Char, Varchar and String.

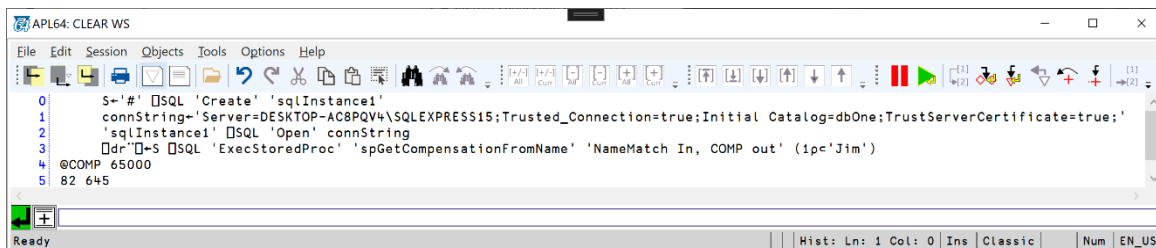
The result is an APL 2-column matrix with one row for each out and in-out parameter:

- Column #1: Parameter name
- Column #2: Parameter value

In and Out parameters with the In parameter specified before the Out parameter:

```
S←'#' SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' SQL 'Open' connString
dr←S SQL 'ExecStoredProc' 'spGetCompensationFromName' 'NameMatch In, COMP
Out' (1p<'Jim')
```



In and Out parameters with the Out parameter specified before the In parameter:

```

S←'#' SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' SQL 'Open' connString
dr←S SQL 'ExecStoredProc' 'spGetCompensationFromName' 'COMP out, NameMatch In'
(1p<'Jim')

```

```

0 S←'#' SQL 'Create' 'sqlInstance1'
1 connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;Trus
2 'sqlInstance1' SQL 'Open' connString
3 dr←S SQL 'ExecStoredProc' 'spGetCompensationFromName' 'COMP out, NameMatch In' (1p<'Jim')
4 @COMP 65000
5 82 645
6

```

Ready | Hist: Ln: 6 Col: 1 | Ins | Classic | Num | EN_US

The stored procedure definition:

```

CREATE PROCEDURE [dbo].[spGetCompensationFromName]
-- Add the parameters for the stored procedure here
    @NameMatch nvarchar(45),
    @COMP float OUTPUT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT @COMP = Compensation from tbOne WHERE Name = @NameMatch
    Select @COMP
END

```

ExecStoredProcCmd

This action can be used to run a stored procedure which has no input or output parameters. The single action argument is an APL text vector (cmdText) containing the stored procedure name. The result is an empty, 2-column APL matrix.

```
res←SQL 'ExecStoredProcCmd' 'uspStoredProc1'
```

```

0 S←'#' □SQL 'Create' 'sqlInstance1'
1 connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Cor
2 'sqlInstance1' □SQL 'Open' connString
3 res←S □SQL 'ExecStoredProcCmd' 'uspStoredProc1'
4 pres
5 0 0
6 |

```

Ready | Hist: Ln: 6 Col: 6 | Ins | Classic | Num | EN_US

GetAllRecords

This action applies to an ☐ SQL instance. The 'GetAllRecords' action will get the remaining records which have not already been read from the specified record set. This action closes the specified record set, because all records have been read.

An exception may occur if the data in the record set has no representation in APL64. In this case, the query which created the record set must be modified, e.g. Cast(), to make the returned data conformable with APL64.

An exception will occur if there are no remaining records to be read.

```

S←'#' □SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' □SQL 'Open' connString
drInt←'sqlInstance1' □SQL 'ExecSelectQuery' 'Select * from tbOne'
x←'sqlInstance1' □sql 'GetAllRecords' drInt

```

```

0 S←'#' □SQL 'Create' 'sqlInstance1'
1 connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
2 'sqlInstance1' □SQL 'Open' connString
3 drInt←'sqlInstance1' □SQL 'ExecSelectQuery' 'Select * from tbOne'
4 x←'sqlInstance1' □sql 'GetAllRecords' drInt
5
6 DOMAIN ERROR: □SQL: Non-APL data in Record# 102 Element# 0 Record type: System.Int64 has no APL representation.
7 [imm:5] x←'sqlInstance1' □sql 'GetAllRecords' drInt
8
9

```

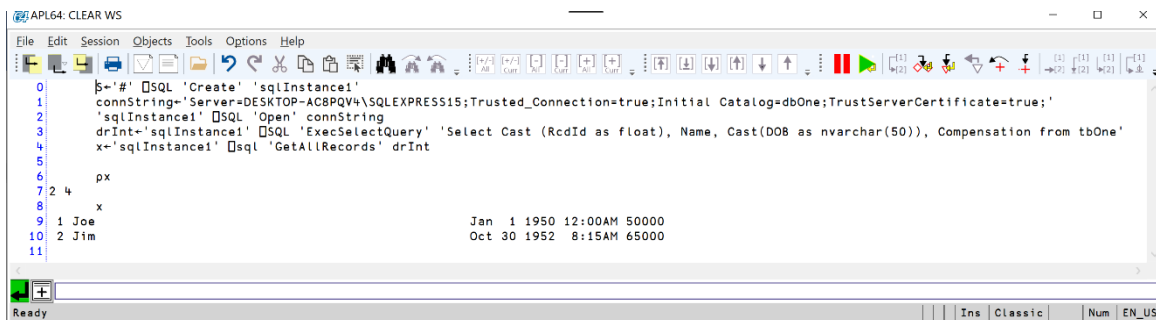
Ready | Hist: Ln: 0 Col: 0 | Ins | Classic | Num | EN_US

In the sample database this particular query failed because the data types being returned in the record set have no representation in APL. Recall that APL provides double, string, integer and bool data types, whereas Microsoft SQL Server includes many additional data types.

To be effective, the SQL Select statement will need to be more sophisticated.

```
S←'#' □SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' □SQL 'Open' connString
drInt←'sqlInstance1' □SQL 'ExecSelectQuery' 'Select Cast (RcdId as float), Name, Cast(DOB
as nvarchar(50)), Compensation from tbOne'
x←'sqlInstance1' □sql 'GetAllRecords' drInt
```



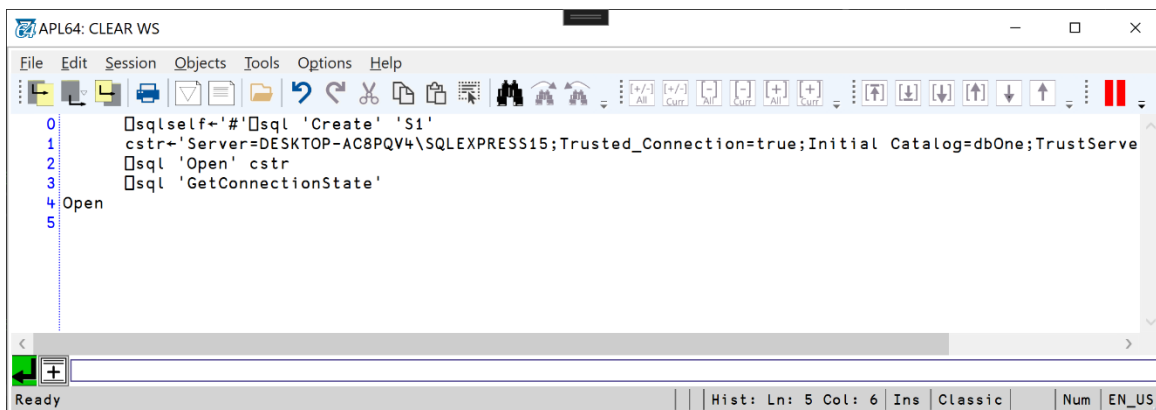
```
APL64: CLEAR WS
File Edit Session Objects Tools Options Help
0  S←'#' □SQL 'Create' 'sqlInstance1'
1  connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
2  'sqlInstance1' □SQL 'Open' connString
3  drInt←'sqlInstance1' □SQL 'ExecSelectQuery' 'Select Cast (RcdId as float), Name, Cast(DOB as nvarchar(50)), Compensation from tbOne'
4  x←'sqlInstance1' □sql 'GetAllRecords' drInt
5
6  px
7  2 4
8  x
9  1 Joe Jan 1 1950 12:00AM 50000
10 2 Jim Oct 30 1952 8:15AM 65000
11
Ready | | Ins | Classic | Num | EN_US
```

GetConnectionState

This action applies to an □SQL instance. The result of the GetConnectionState action is a text vector indicating the connection state of the specified □SQL instance to an Sql database.

```
□sqlself←'#' □sql 'Create' 'S1'
cstr←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

□sql 'Open' cstr
□sql 'GetConnectionState'
```



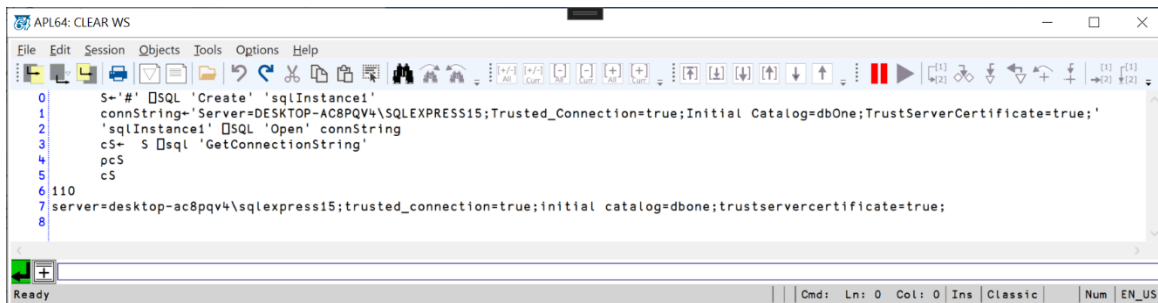
```
APL64: CLEAR WS
File Edit Session Objects Tools Options Help
0  □sqlself←'#' □sql 'Create' 'S1'
1  cstr←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial Catalog=dbOne;TrustServe
2  □sql 'Open' cstr
3  □sql 'GetConnectionState'
4  Open
5
Ready | | Hist: Ln: 5 Col: 6 | Ins | Classic | Num | EN_US
```

GetConnectionString

This action applies to an ☐ SQL instance. The result of the GetConnectionString is the Sql connection string, if any, associated with the specified ☐ SQL instance. Use the Open action to set the connection string.

```
S←'#' ☐ SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' ☐ SQL 'Open' connString
cS← S ☐ sql 'GetConnectionString'
pcS
cS
```



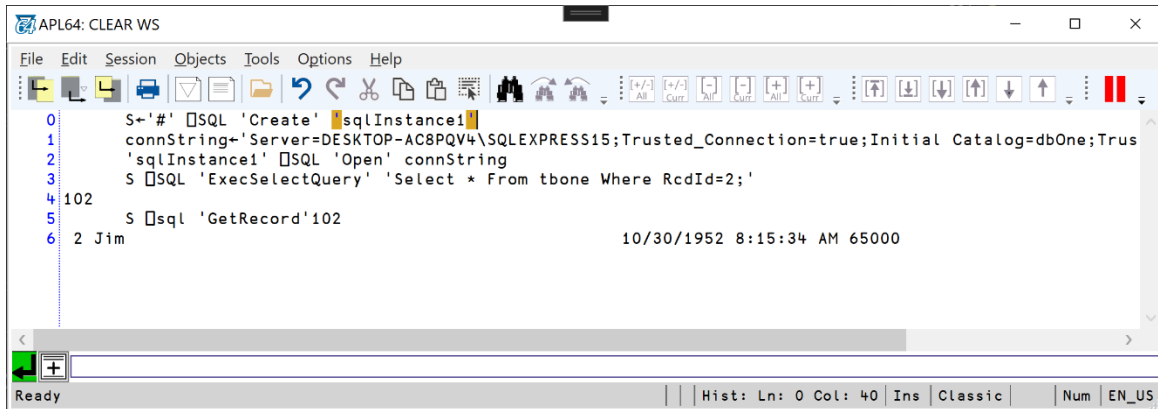
GetRecord

This action applies to an ☐ SQL instance. The right argument is the ☐ SQL record set number. The GetRecord action will read the next available record, if any, from the specified record set and return its values to the APL64 instance.

An exception may occur if the data in the record set has no representation in APL64. In this case the query which created the record set must be modified to make the returned data conformable with APL64. If such an exception does occur, the Cast() method of the database can generally convert the source data in the database record to an APL64 data type.

```
S←'#' ☐ SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' ☐ SQL 'Open' connString
S ☐ SQL 'ExecSelectQuery' 'Select * From tbone Where RcdId=2;'
S ☐ sql 'GetRecord' 102
```



GetRecords

Synonym: GetRcds

Syntax: matrix $\leftarrow$ S SQL 'GetRcds' rSn nRcds

rSn is the record set number

nRcds is the number of records to return

This action applies to an SQL instance. The GetRecords action will return a matrix of the requested number of records from the record set, up to the number of remaining records which have not already been read from the record set.

The record set will be closed if all records have been read.

An exception may occur if the data in the record set has no representation in APL64. In this case the query which created the record set must be modified to make the returned data conformable with APL64. If such an exception does occur, the Cast() method of the database can generally convert the source data in the database record to an APL64 data type.

```

S←'#' SQL 'Create' 'sql1'
CS←'Data Source=5760-68TCHL3;Trusted_Connection=true;Integrated
Security=True;TrustServerCertificate=True'
S SQL 'Open' CS
rsn←S SQL 'ExecSelect' 'Select * FROM dbOne.dbo.tbOne'
p←S SQL 'GetAllRcds' rsn
rsn←S SQL 'ExecSelect' 'Select * FROM dbOne.dbo.tbOne'
p←S SQL 'GetRcds' rsn 5

```

The screenshot shows the APL64: CLEAR WS interface. The main window displays the following SQL code and its results:

```

0 S←'#'SQL 'Create' 'sql1'
1 CS←'Data Source=5760-68TCHL3;Trusted_Connection=true;Integrated Security=True;TrustServerCertificate=True'
2 SSQL 'Open' CS
3 rsn←SSQL 'ExecSelect' 'Select * FROM dbOne.dbo.tbOne'
4 p←SSQL 'GetAllRcds' rsn
5 1 Salmon, A. 1/1/2000 12:00:00 AM 100000
6 2 Trout,Brook DBNull DBNull
7 3 Trout,Brown DBNull DBNull
8 3 4
9 rsn←SSQL 'ExecSelect' 'Select * FROM dbOne.dbo.tbOne'
10 p←SSQL 'GetRcds' rsn 5
11 1 Salmon, A. 1/1/2000 12:00:00 AM 100000
12 2 Trout,Brook DBNull DBNull
13 3 Trout,Brown DBNull DBNull
14 3 4
15

```

The status bar at the bottom indicates: Ready | Hist: Ln: 15 Col: 6 | Ins | Classic | Num | EN_US

GroupSeparator

This action returns the [.Net Number Group Separator](#) text applicable to the specified ☐SQL instance.

- ☐SQLSelf←'#'SQL 'Create' 'S'
 - ☐SQL 'DateFormat'
 - ☐SQL 'DecimalSeparator'
 - ☐SQL 'GroupSeparator'

The screenshot shows the APL64: CLEAR WS interface. The main window displays the following SQL code:

```

0 SQLSelf←'#'SQL 'Create' 'S'
1 SQL 'DateFormat'
2 M/d/yyyy
3 SQL 'DecimalSeparator'
4 .
5 SQL 'GroupSeparator'
6 ,
7

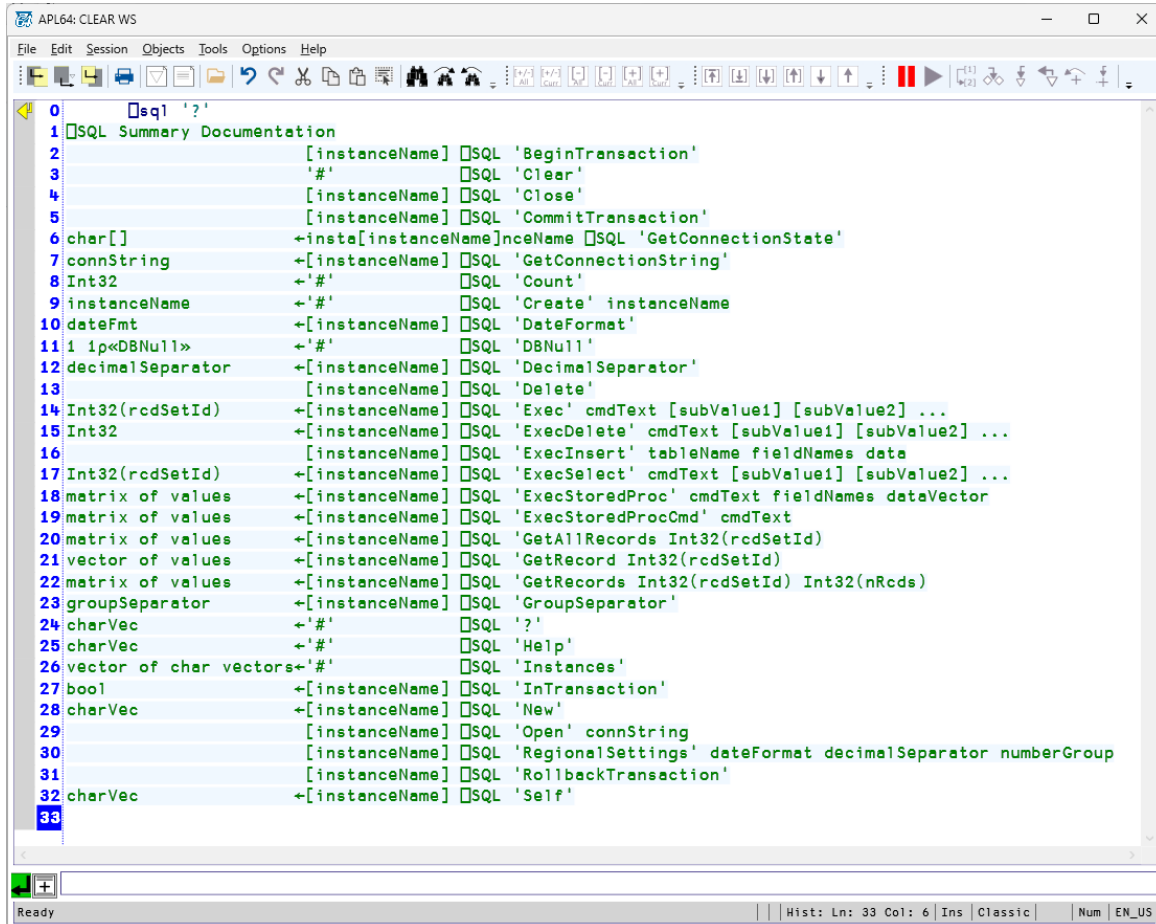
```

The status bar at the bottom indicates: Ready | Hist: Ln: 3 Col: 29 | Ins | Classic | Num | EN_US

? or Help

This action is performed on the `SQL` object. The `?` and `Help` actions will return a text array containing a summary of the `SQL` syntax.

```
SQL '?'
```

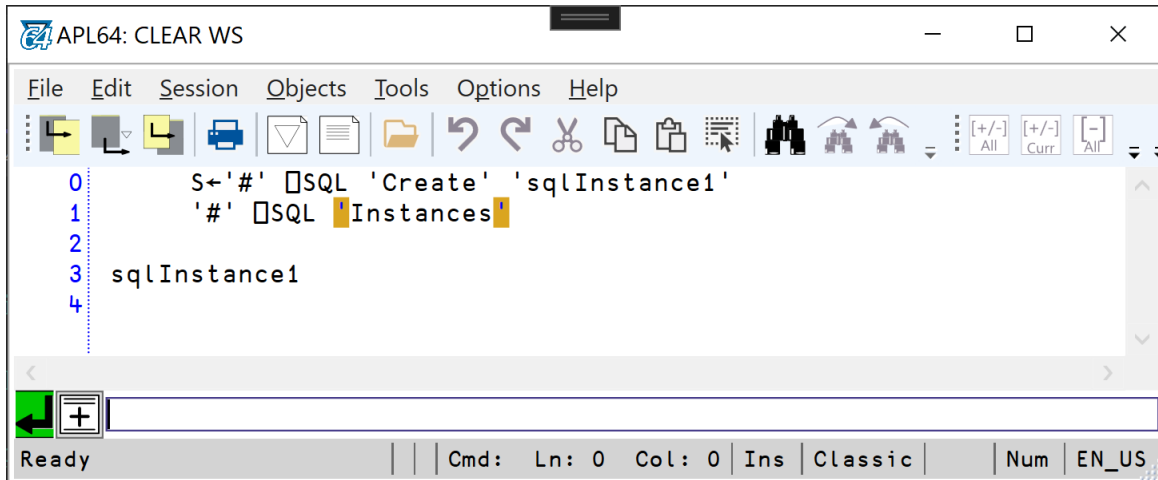


```
0: SQL '?'
1: SQL Summary Documentation
2: [instanceName] SQL 'BeginTransaction'
3: '#' SQL 'Clear'
4: [instanceName] SQL 'Close'
5: [instanceName] SQL 'CommitTransaction'
6: char[] ←insta[instanceName]nceName SQL 'GetConnectionState'
7: connString ←[instanceName] SQL 'GetConnectionString'
8: Int32 ← '#' SQL 'Count'
9: instanceName ← '#' SQL 'Create' instanceName
10: dateFmt ←[instanceName] SQL 'DateFormat'
11: 1p«DBNull» ← '#' SQL 'DBNull'
12: decimalSeparator ←[instanceName] SQL 'DecimalSeparator'
13: [instanceName] SQL 'Delete'
14: Int32(rcdSetId) ←[instanceName] SQL 'Exec' cmdText [subValue1] [subValue2] ...
15: Int32 ←[instanceName] SQL 'ExecDelete' cmdText [subValue1] [subValue2] ...
16: [instanceName] SQL 'ExecInsert' tableName fieldNames data
17: Int32(rcdSetId) ←[instanceName] SQL 'ExecSelect' cmdText [subValue1] [subValue2] ...
18: matrix of values ←[instanceName] SQL 'ExecStoredProc' cmdText fieldNames dataVector
19: matrix of values ←[instanceName] SQL 'ExecStoredProcCmd' cmdText
20: matrix of values ←[instanceName] SQL 'GetAllRecords' Int32(rcdSetId)
21: vector of values ←[instanceName] SQL 'GetRecord' Int32(rcdSetId)
22: matrix of values ←[instanceName] SQL 'GetRecords' Int32(rcdSetId) Int32(nRcds)
23: groupSeparator ←[instanceName] SQL 'GroupSeparator'
24: charVec ← '#' SQL '?'
25: charVec ← '#' SQL 'Help'
26: vector of char vectors ← '#' SQL 'Instances'
27: bool ←[instanceName] SQL 'InTransaction'
28: charVec ←[instanceName] SQL 'New'
29: [instanceName] SQL 'Open' connString
30: [instanceName] SQL 'RegionalSettings' dateFormat decimalSeparator numberGroup
31: [instanceName] SQL 'RollbackTransaction'
32: charVec ←[instanceName] SQL 'Self'
33:
```

Instances

This action is performed on the `SQL` object. The result is a vector of char vectors representing a list of instance names.

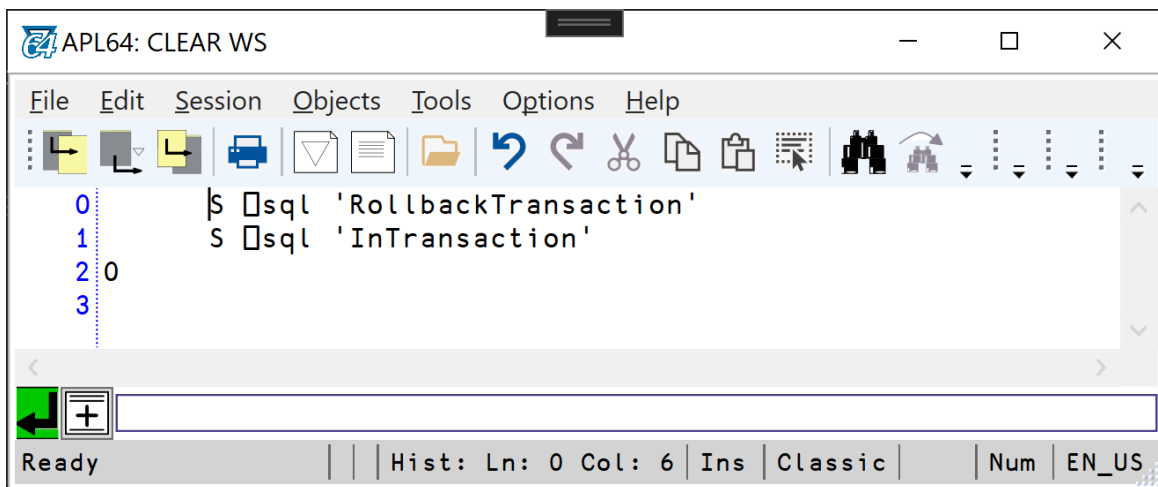
```
S←'#' SQL 'Create' 'sqlInstance1'
'#' SQL 'Instances'
```



InTransaction

This action applies to an ☐SQL instance. The Boolean result of the InTransaction action indicates if a transaction block is in existence for the specified ☐SQL instance.

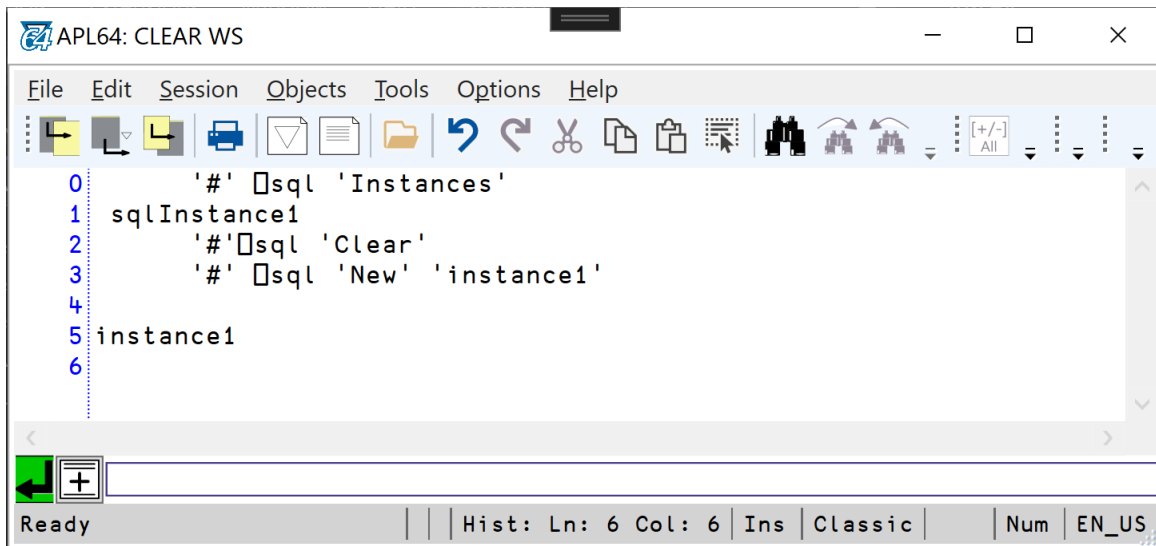
☐ SQL 'RollbackTransaction'
☐ SQL 'InTransaction'



New

This action is performed on the ☐SQL object. The New action will create an ☐SQL instance with a user-provided name in the right argument. Multiple ☐SQL instances are possible in the same APL64 instance, so that multiple Sql databases may be conveniently accessed. The New action will fail if the named instance already exists. The New action for a particular Sql database is generally used once in an APL64 instance. The New action does not open a connection to an Sql database, use the Open action for that purpose. The result of a successful 'New' action is a text vector containing the ☐SQL instance name.

'#' ☐ SQL 'Clear'
'#' ☐ SQL 'New' 'instance1'

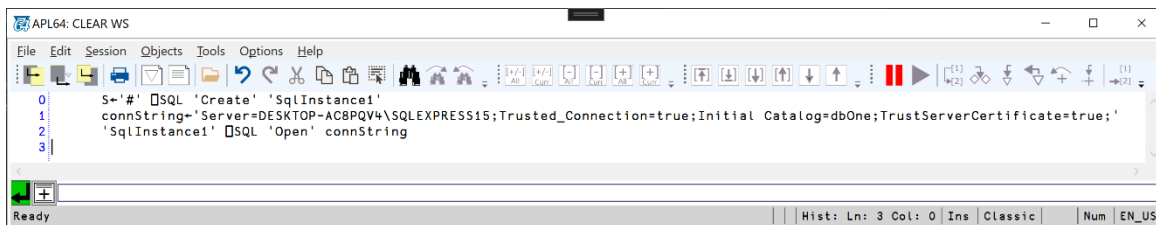


Open

This action applies to an ☐ SQL instance. The Open action requires the specification of a [connection string](#) as the right argument.

```
S←'#' ☐ SQL 'Create' 'SqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'SqlInstance1' ☐ SQL 'Open' connString
```



RegionalSettings

- This action can be used to set the regional values for the specified ☐ SQL instance:
- ShortDatePattern
- CurrencyDecimalSeparator, NumberDecimalSeparator & PercentDecimalSeparator
- CurrencyGroupSeparator, NumberGroupSeparator & PercentGroupSeparator

```
'instance1' ☐ sql 'RegionalSettings' 'M/d/yyyy' '.' ','
```

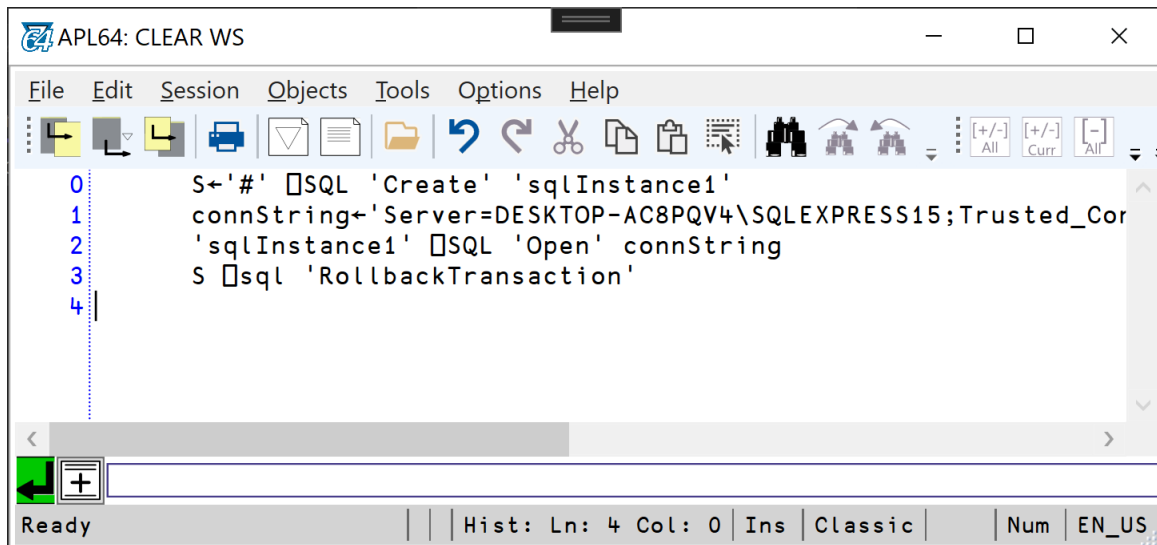
RollbackTransaction

This action applies to an ☐ SQL instance. The RollbackTransaction action is used to cancel any pending SQL operations which are included in the current SQL transaction. The

RollbackTransaction action has no result. The RollbackTransaction action has no effect if no SQL transaction is in progress.

```
S←'#' SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' SQL 'Open' connString
S SQL 'RollbackTransaction'
```



```
S←'#' SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' SQL 'Open' connString
S SQL 'ExecSelectQuery' 'Select Name from tbone;'
S SQL 'GetAllRecords' 102
S SQL 'BeginTransaction'
S SQL 'ExecInsertQuery' 'tbone' 'RcdId,Name,DOB,Compensation' (3 'Salmon' '1966-01-01'
90500.56)
S SQL 'RollbackTransaction'
S SQL 'ExecSelectQuery' 'Select Name from tbone;'
S SQL 'GetAllRecords' 103
S SQL 'ExecDeleteQuery' 'Delete From tbone Where RcdId=3;'
```

The screenshot shows the APL64: CLEAR WS window with a menu bar (File, Edit, Session, Objects, Tools, Options, Help) and a toolbar. The main text area contains the following SQL commands and their results:

```

0 S←'#' □SQL 'Create' 'sqlInstance1'
1 connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Con
2 'sqlInstance1' □SQL 'Open' connString
3 S □SQL 'ExecSelectQuery' 'Select Name from tbone;'
4 S □SQL 'GetAllRecords' 102
5 S □SQL 'BeginTransaction'
6 S □SQL 'ExecInsertQuery' 'tbone' 'RcdId,Name,DOB,Compensati
7 S □SQL 'RollbackTransaction'
8 S □SQL 'ExecSelectQuery' 'Select Name from tbone;'
9 S □SQL 'GetAllRecords' 103
10 S □SQL 'ExecDeleteQuery' 'Delete From tbone Where RcdId=3;'
11 102|
12 Joe
13 Jim
14 103
15 Joe
16 Jim
17 0
18

```

At the bottom, there is a status bar showing 'Ready' and a history bar with 'Hist: Ln: 11 Col: 3 | Ins | Classic | Num | EN_US'.

Self

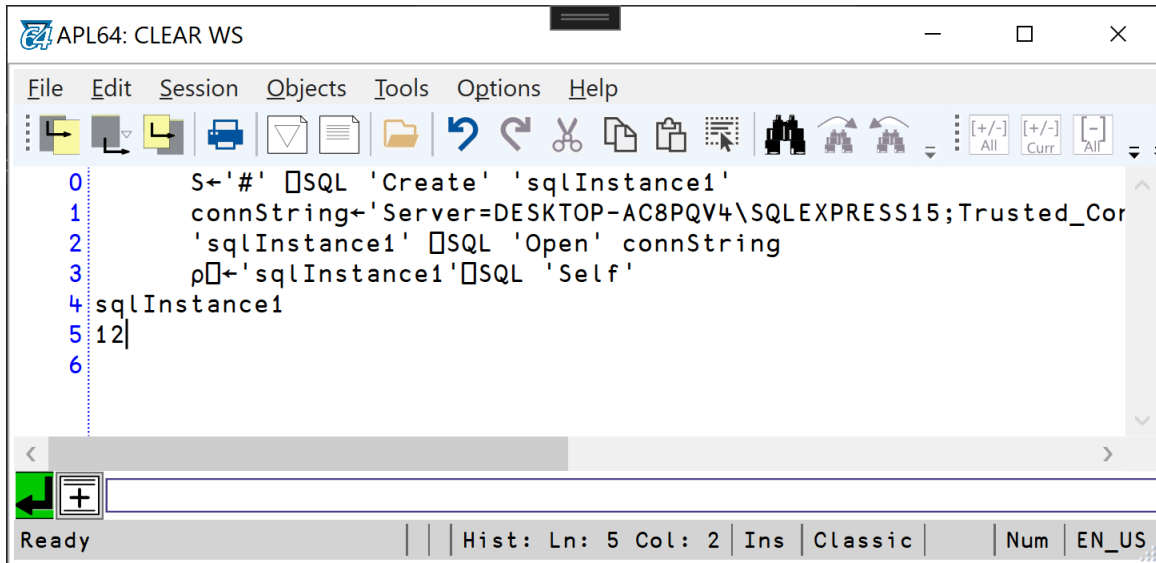
This action is performed on the ☐SQL object. The right argument specifies an ☐SQL instance name for a potentially existing ☐SQL instance. If the ☐SQL instance exists in the APL64 instance, the result of the Self action is the ☐SQL instance name.

```

S←'#' □SQL 'Create' 'sqlInstance1'
connString←'Server=DESKTOP-AC8PQV4\SQLEXPRESS15;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

'sqlInstance1' □SQL 'Open' connString
ρ□←'sqlInstance1'□SQL 'Self'

```



Learn Structured Query Language

<http://www.w3schools.com/sql/default.asp>

Some Additional Examples

Create Table

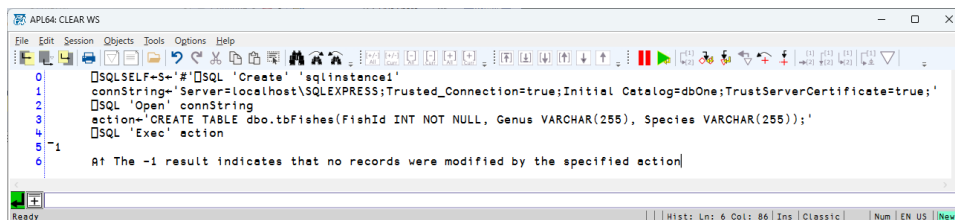
Use the ☐SQL 'Exec' action to create a new table within an existing database using an [appropriate SQL statement](#).

```

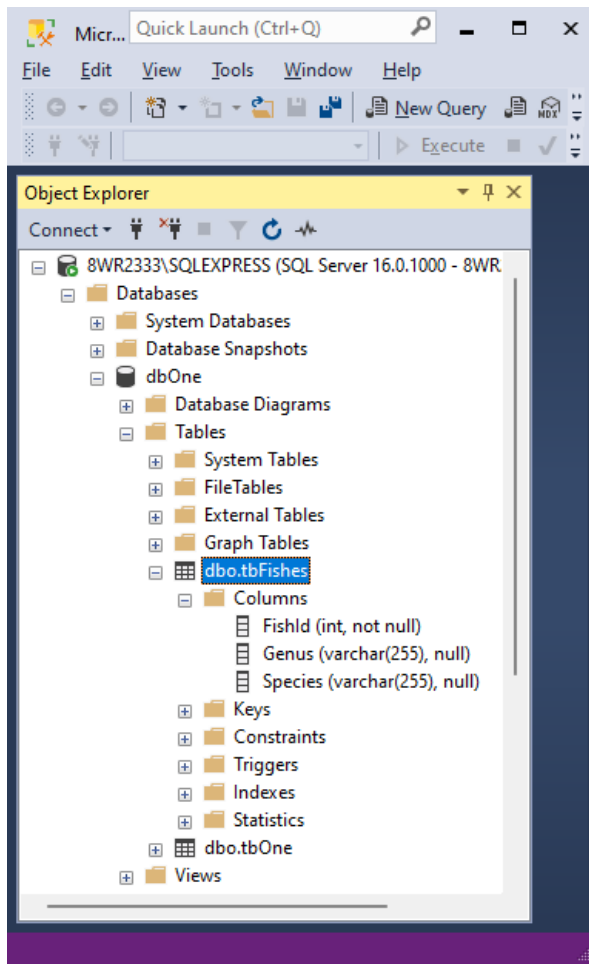
☐SQLSELF←S←'#'☐SQL 'Create' 'sqlInstance1'
connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;'

☐SQL 'Open' connString
action←'CREATE TABLE dbo.tbFishes(FishId INT NOT NULL, Genus VARCHAR(255), Species
VARCHAR(255));'
☐SQL 'Exec' action

```



Verify that the table was created using Microsoft SQL Server Management Studio:



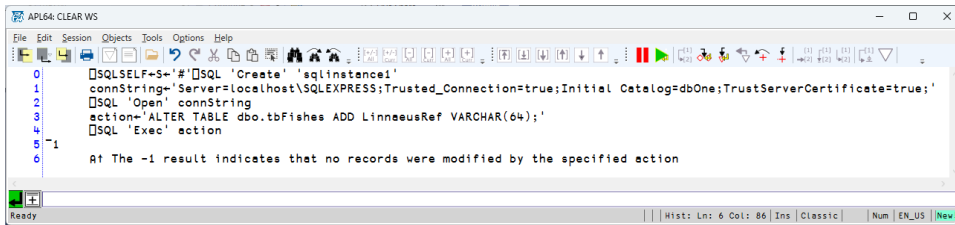
Alter Table

Use the ☐ SQL 'Exec' action using an Alter Table SQL statement to modify an existing table in an existing database. In this example, a new column is added to an existing table using an [appropriate SQL statement](#).

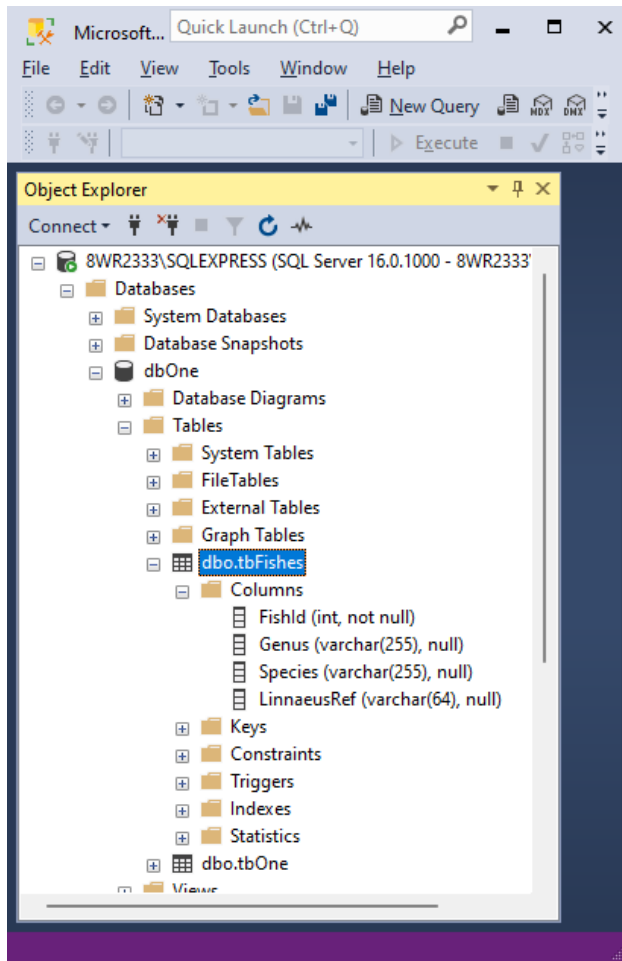
☐ SQLSELF←S←'#'☐ SQL 'Create' 'sqlinstance1'
 connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial
 Catalog=dbOne;TrustServerCertificate=true;'

☐ SQL 'Open' connString
 action←'ALTER TABLE dbo.tbFishes ADD LinnaeusRef VARCHAR(64);'

☐ SQL 'Exec' action



Verify that the column was added using Microsoft SQL Server Management Studio:



Create/Alter Procedure

This example uses the tbFishes table created in the previous Create Table and Alster Table examples.

Use the ☐ SQL 'Exec' action using a Create Procedure or Alter Procedure statement to create or alter a stored procedure. In this example a new stored procedure is added to an existing database using an [appropriate SQL statement](#):

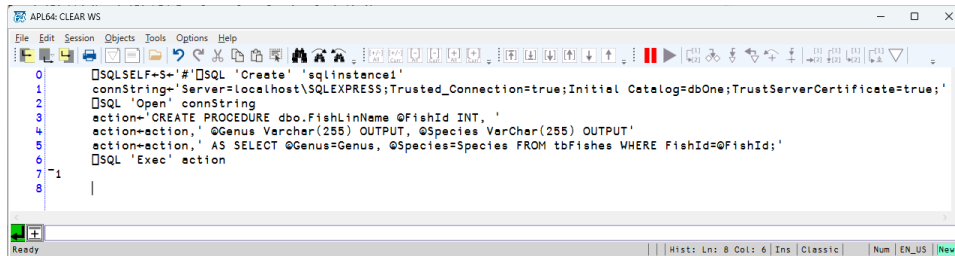
☐SQLSELF←S←'#'[SQL 'Create' 'sqlinstance1'

```
connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial  
Catalog=dbOne;TrustServerCertificate=true;'
```

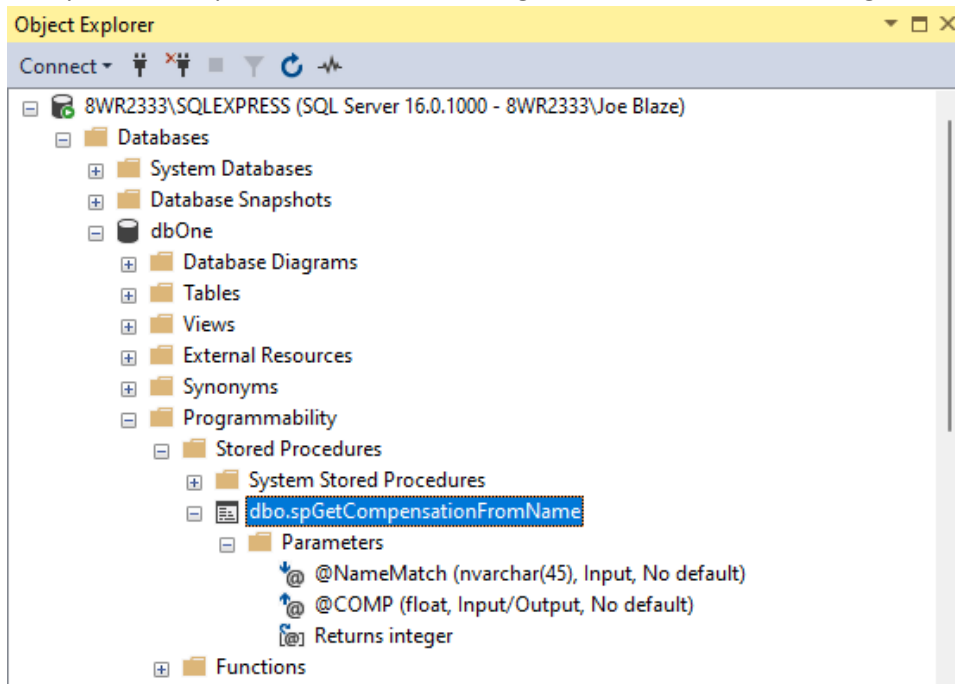
☐ SQL 'Open' connString

```
action←'CREATE PROCEDURE dbo.FishLinName @FishId INT, '  
action←action,' @Genus Varchar(255) OUTPUT, @Species VarChar(255) OUTPUT'  
action←action,' AS SELECT @Genus=Genus, @Species=Species FROM tbFishes WHERE  
FishId=@FishId;'
```

☐ SQL 'Exec' action



Verify the stored procedure is created using Microsoft SQL Server Management Studio:



Add a data record to the tbFishes table:

```
☐ SQLSELF←S←'#'[SQL 'Create' 'sqlinstance1'  
connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial  
Catalog=dbOne;TrustServerCertificate=true;'
```

☐ SQL 'Open' connString

```

SQL 'ExecInsertQuery' 'tbFishes' 'FishId, Genus, Species, LinnaeusRef' (1 4 p 4 'Sander'
'lucioperca' 'Perca lucioperca')

```

```

APL64: CLEAR WS
File Edit Session Objects Tools Options Help
0  [SQLSELF=S←'#'[SQL 'Create' 'sqlinstance1'
1  connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;';
2  [SQL 'Open' connString
3  [SQL 'ExecInsertQuery' 'tbFishes' 'FishId, Genus, Species, LinnaeusRef' (1 4 p 4 'Sander' 'lucioperca' 'Perca lucioperca')
Ready

```

Verify the data record was inserted using Microsoft SQL Server Management Studio:

SQLQuery28.sql - 8...333\Joe Blaze (66)

```

SELECT TOP (1000) [FishId]
, [Genus]
, [Species]
, [LinnaeusRef]
FROM [dbOne].[dbo].[tbFishes]

```

FishId	Genus	Species	LinnaeusRef
1	4	Sander	lucioperca

Query executed... 8WR2333\SQLEXPRESS (16.0 RTM) 8WR2333\Joe Blaze (66) dbOne 00:00:00 1 rows

Run the stored procedure in APL64:

```

SQLSELF←S←'#'[SQL 'Create' 'sqlinstance1'
connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial
Catalog=dbOne;TrustServerCertificate=true;';

[SQL 'Open' connString
dr←[SQL 'ExecStoredProc' 'FishLinName' 'FishId In, Genus Out, Species Out' (1p4)

```

```

APL64: CLEAR WS
File Edit Session Objects Tools Options Help
0  [SQLSELF=S←'#'[SQL 'Create' 'sqlinstance1'
1  connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;';
2  [SQL 'Open' connString
3  [SQL 'ExecStoredProc' 'FishLinName' 'FishId In, Genus Out, Species Out' (1p4)
4
5  @Genus Sander
6  @Species lucioperca
7  82 82
8  82 82
9
Ready

```

Update Table Record

This example uses the tbFishes table created in the previous Create Table, Alter Table and Create/Alter Procedure examples.

Method #1:

Use the ☐ SQL 'Exec' action with an [appropriate SQL UPDATE statement](#):

- ☐ SQLSELF←S←'#'☐ SQL 'Create' 'sqlinstance1'
connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
- ☐ SQL 'Open' connString
- ☐ SQL 'Exec' 'UPDATE tbFishes SET FishId=5 WHERE FishId=4;' ☐ Modify a column value for a specific record
- ☐ SQL 'ExecSelectQuery' 'Select * FROM tbFishes' ☐ Check the modification was made
- ☐ SQL 'GetAllRecords' 102
- ☐ SQL 'Exec' 'UPDATE tbFishes SET FishId=4 WHERE FishId=5;' ☐ Reverse the modification
- ☐ SQL 'ExecSelectQuery' 'Select * FROM tbFishes' ☐ Check the reversal was made
- ☐ SQL 'GetAllRecords' 103

```
0  SQLSELF←S←'#'☐ SQL 'Create' 'sqlinstance1'  
1  connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'  
2  ☐ SQL 'Open' connString  
3  ☐ SQL 'Exec' 'UPDATE tbFishes SET FishId=5 WHERE FishId=4;'  
4  1  
5  ☐ SQL 'ExecSelectQuery' 'Select * FROM tbFishes'  
6  102  
7  ☐ SQL 'GetAllRecords' 102  
8  4 Sander lucioiperca Perca locioiperca  
9  ☐ SQL 'Exec' 'UPDATE tbFishes SET FishId=4 WHERE FishId=5;'  
10 0  
11 ☐ SQL 'ExecSelectQuery' 'Select * FROM tbFishes'  
12 103  
13 ☐ SQL 'GetAllRecords' 103  
14 4 Sander lucioiperca Perca locioiperca  
15
```

Method #2:

Within an SQL transaction, delete the desired table record and insert the modified table record

- ☐ SQLSELF←S←'#'☐ SQL 'Create' 'sqlinstance1'
connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
- ☐ SQL 'Open' connString
- ☐ SQL 'BeginTransaction'
- ☐ SQL 'ExecDeleteQuery' 'Delete From tbFishes Where FishId=4;'
- ☐ SQL 'ExecInsertQuery' 'tbFishes' 'FishId, Genus, Species, LinnaeusRef' (1 4 p6 'XSander' 'Xlucioiperca' 'Perca locioiperca')
- ☐ SQL 'CommitTransaction'
- ☐ SQL 'ExecSelectQuery' 'Select * FROM tbFishes' ☐ Check the modification was made
- ☐ SQL 'GetAllRecords' 102

```

0  [SQLSELF←S←'#'[SQL 'Create' 'sqlinstance1'
1  connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
2  [SQL 'Open' connString
3  [SQL 'BeginTransaction'
4  [SQL 'ExecDeleteQuery' 'Delete From tbFishes Where FishId=4;'
5  [SQL 'ExecInsertQuery' 'tbFishes' 'FishId, Genus, Species, LinnaeusRef' (1 4 p4 'XSender' 'Xlucioperca' 'Perca locio
6  [SQL 'CommitTransaction'
7  [SQL 'ExecSelectQuery' 'Select * FROM tbFishes'      A Check the modification was made
8  [SQL 'GetAllRecords' 102
9  1
10 102
11 6 XSender Xlucioperca Perca locio
12

```

After completing this example, restore the tbFishes record state:

- ☐ SQLSELF←S←'#'[SQL 'Create' 'sqlinstance1'
- connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
- ☐ SQL 'Open' connString
- ☐ SQL 'BeginTransaction'
- ☐ SQL 'ExecDeleteQuery' 'Delete From tbFishes Where FishId=6;'
- ☐ SQL 'ExecInsertQuery' 'tbFishes' 'FishId, Genus, Species, LinnaeusRef' (1 4 p4 'Sander' 'lucioperca' 'Perca locio
- ☐ SQL 'CommitTransaction'
- ☐ SQL 'ExecSelectQuery' 'Select * FROM tbFishes' ☞ Check the modification was made
- ☐ SQL 'GetAllRecords' 102

```

0  [SQLSELF←S←'#'[SQL 'Create' 'sqlinstance1'
1  connString←'Server=localhost\SQLEXPRESS;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
2  [SQL 'Open' connString
3  [SQL 'BeginTransaction'
4  [SQL 'ExecDeleteQuery' 'Delete From tbFishes Where FishId=6;'
5  [SQL 'ExecInsertQuery' 'tbFishes' 'FishId, Genus, Species, LinnaeusRef' (1 4 p4 'Sander' 'lucioperca' 'Perca locio
6  [SQL 'CommitTransaction'
7  [SQL 'ExecSelectQuery' 'Select * FROM tbFishes'      A Check the modification was made
8  [SQL 'GetAllRecords' 102
9  1
10 102
11 4 Sander lucioperca Perca locio
12

```

Obtain Columns Information for a Table

SQL queries can be made to 'built-in' tables in the database, e.g. [obtain column information](#) for a programmer-created table:

- ☐ SQLSELF←S←'#'[SQL 'Create' 'sqlinstance1'
- connString←'Server=8WR2333\SQLEXPRESS;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
- ☐ SQL 'Open' connString
- colNamesQuery←"SELECT * FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = N'tbOne'"
- ☐ SQL 'ExecSelectQuery' colNamesQuery

```

colNames←⊞SQL 'GetAllRecords' 102
colNames
pcolNames
p⊞←colNames[;1]
⊞dr colNames

```

The screenshot shows the APL64: CLEAR WS window. The code in the workspace is as follows:

```

0  ⊞SQLSELF+S+'#'⊞SQL 'Create' 'sqlinstance1'
1  connString+'Server=8WR2333\SQLEXPRESS;Trusted_Connection=true;Initial Catalog=dbOne;TrustServerCertificate=true;'
2  ⊞SQL 'Open' connString
3  colNamesQuery+'SELECT * FROM INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = N'tbOne'"
4  ⊞SQL 'ExecSelectQuery' colNamesQuery
5  102
6  colNames←⊞SQL 'GetAllRecords' 102
7  colNames
8  dbOne  dbo  tbOne  RcdId      1      NO  bigint      19 10 0
9
10 dbOne  dbo  tbOne  Name       2      YES nchar    50 100      UNICODE      SQL_Latin1_General_CP1_CI_AS
11
12 dbOne  dbo  tbOne  DOB        3      YES datetime      3
13
14 dbOne  dbo  tbOne  Compensation 4      YES float      53 2
15  pcolNames
16  4 23
17  p⊞←colNames[;1]
18  dbOne  dbOne  dbOne  dbOne|
19  4
20  ⊞dr colNames
21  807
22

```

The execution results for line 7 are displayed below the code:

dbOne	dbo	tbOne	RcdId	1	NO	bigint	19	10	0			
dbOne	dbo	tbOne	Name	2	YES	nchar	50	100		UNICODE	SQL_Latin1_General_CP1_CI_AS	
dbOne	dbo	tbOne	DOB	3	YES	datetime			3			
dbOne	dbo	tbOne	Compensation	4	YES	float	53	2				

The status bar at the bottom indicates: Ready | Hist: Ln: 18 Col: 24 | Ins | Classic | Num | EN_US