Using □ODBC

Contents

Overview

Open Database Connectivity (ODBC) provides a single interface which can be used to access many different relational database implementations.

The APL64 □ODBC system function provides a structured query language (ODBC)-based interface to a relational database. The object model for □ODBC includes methods and properties to easily create a secure, high-performance application system data interface.

The APL64 implementation of □ODBC is cross-platform compatible, but its applicability is mainly suitable for the Windows operating system environment because of limited availability of ODBC drivers for other operating system environments. The ability to use the APL64 □ODBC system function on a specific operating system platform depends on the availability of an x64 ODBC driver for that database and operating system environment.

APL64 does not incorporate ODBC drivers. An ODBC driver is database-specific software produced by database developers for specific operating system environment. It is up to the APL64 programmer to assure that the end user's workstation has access to the applicable ODBC driver. ODBC drivers may not be available for all databases and all operating system environments.

Examples:
There is a Microsoft ODBC driver for Microsoft ODBC Server for Windows, Linux, and macOS, but there is no Microsoft ODBC driver for Microsoft ODBC Server on Android. Third-party providers have published ODBC drivers for Microsoft ODBC Server on various platforms.

There is a Microsoft ODBC driver for Microsoft Access on Windows, but there is no Microsoft ODBC driver for Microsoft Access on Android or macOS. Third-party providers have published ODBC drivers for Microsoft Access on various platforms.

Prerequisites:

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

Alternatives to □ODBC

For some database software there is no alternative to ODBC technology and □ODBC, e.g. Microsoft Access.

Using □ODBC requires that database- and operating system-specific ODBC software (drivers) be installed on the target workstation. Such drivers add an interface layer in the programming stack which can have performance implications and, in some circumstances, employ pre-.Net software.

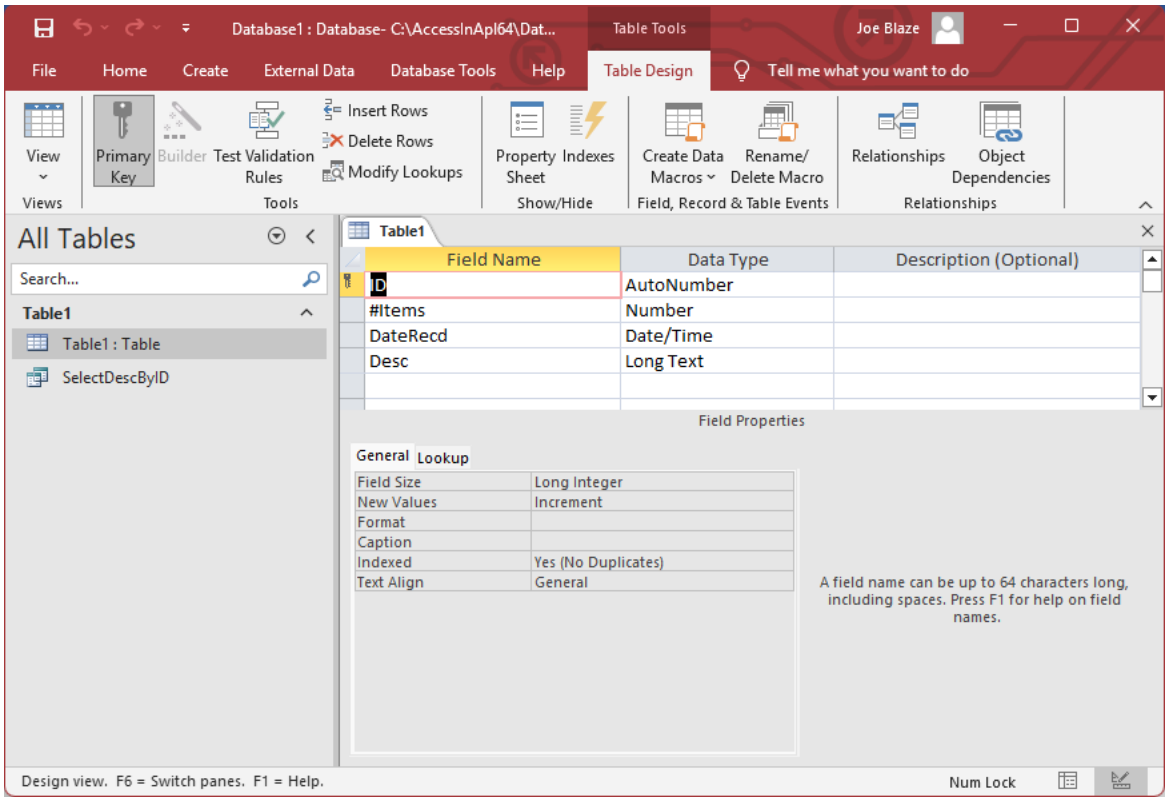
APL64 has system functions for Microsoft SQL Server (□SQL), Sqlite (□Sqlite), MySql (□Sqmy) and IBMDB2 (□SQDB2) which do not require additional driver installation, and have improved performance compared to □ODBC.

No cost Microsoft Access Runtime (Microsoft 365 Access Runtime)

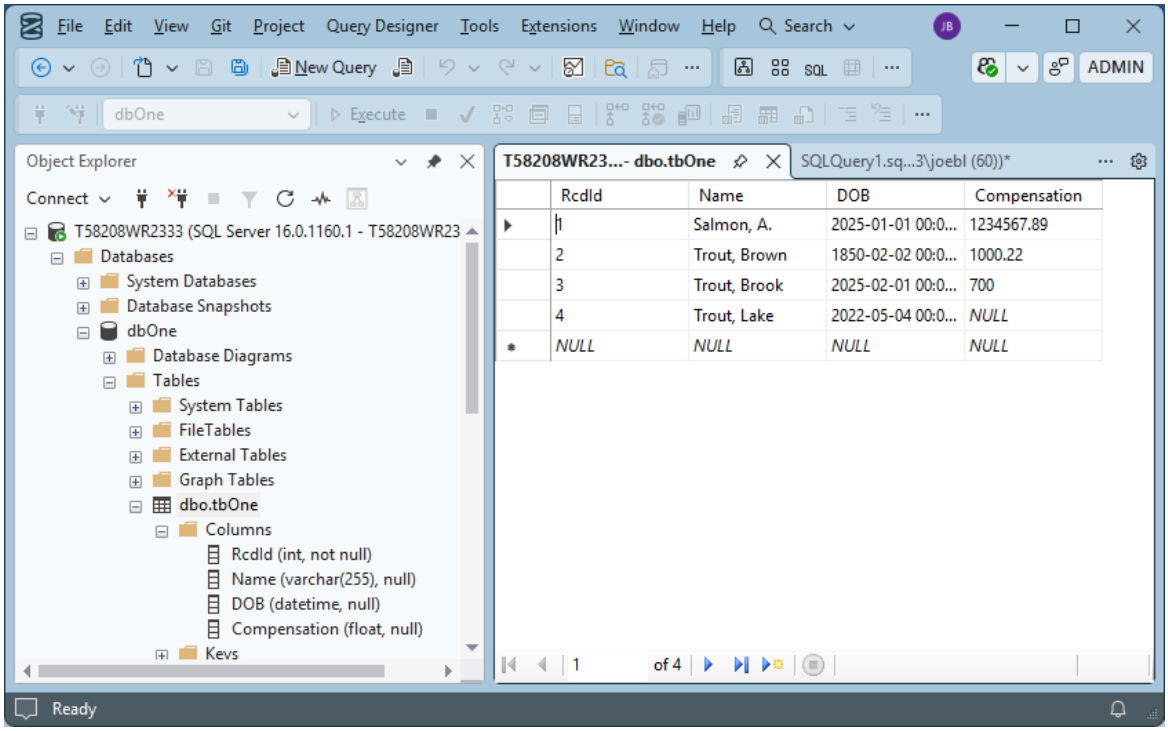
Select the x64 version of the Microsoft 365 Access Runtime for APL64.

The examples in this document assume that a sample ODBC database is installed, running and accessible to the target workstation. The examples in this document also assume that this sample ODBC database has a table with sample columns and sample data.

Microsoft Access sample database Database1, with table Table1

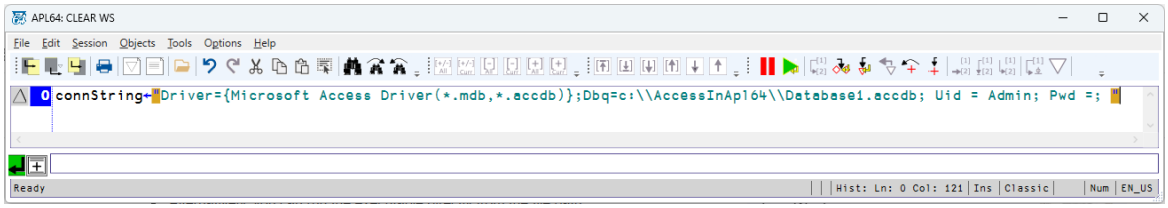


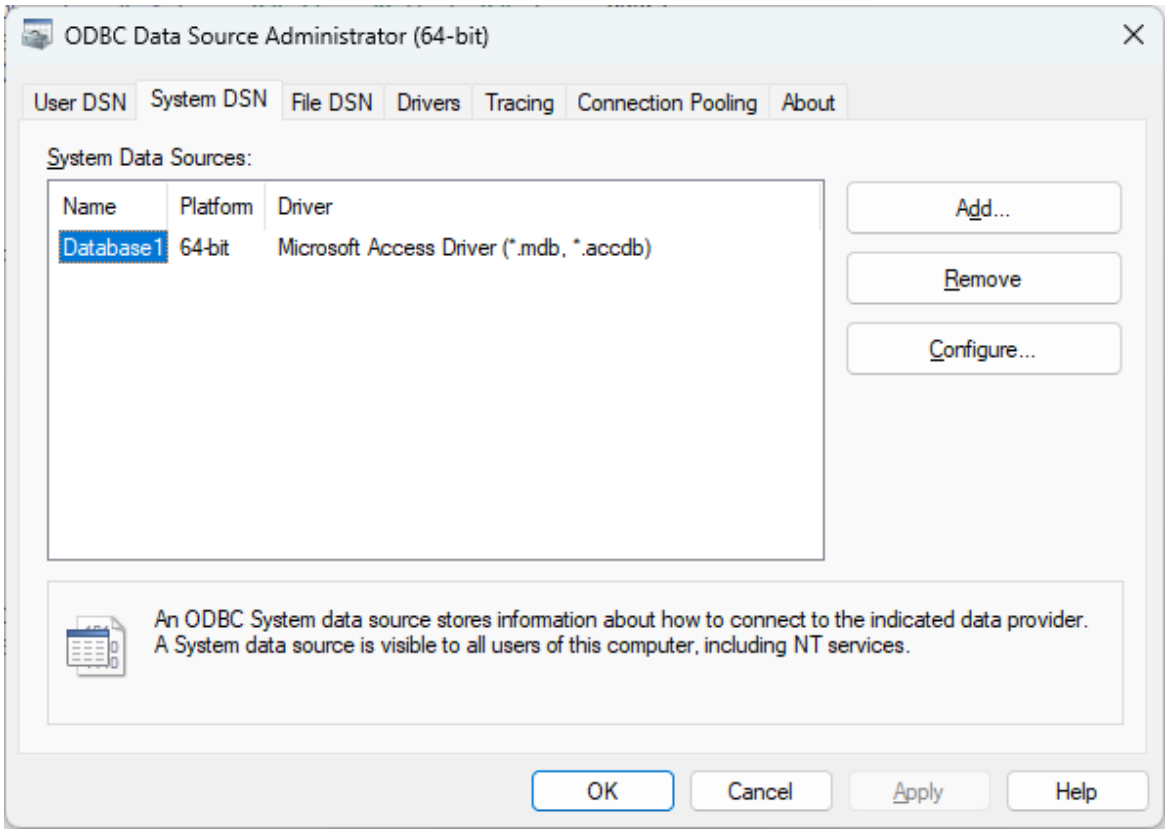
Microsoft SQL Server sample database dbOne with table tbOne

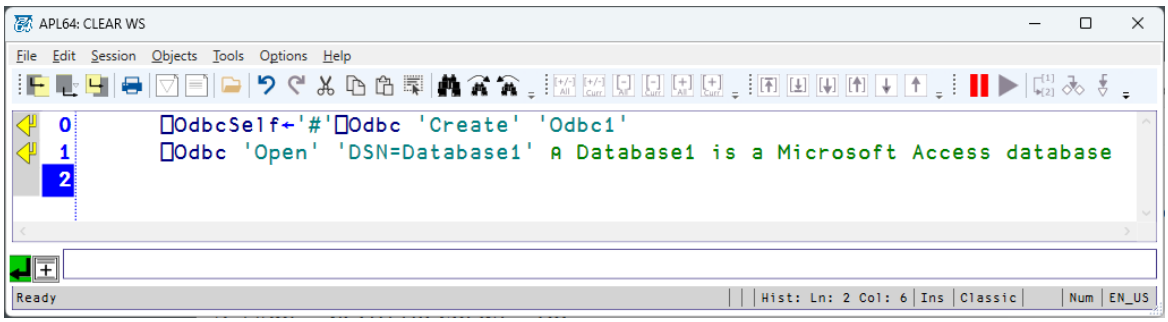


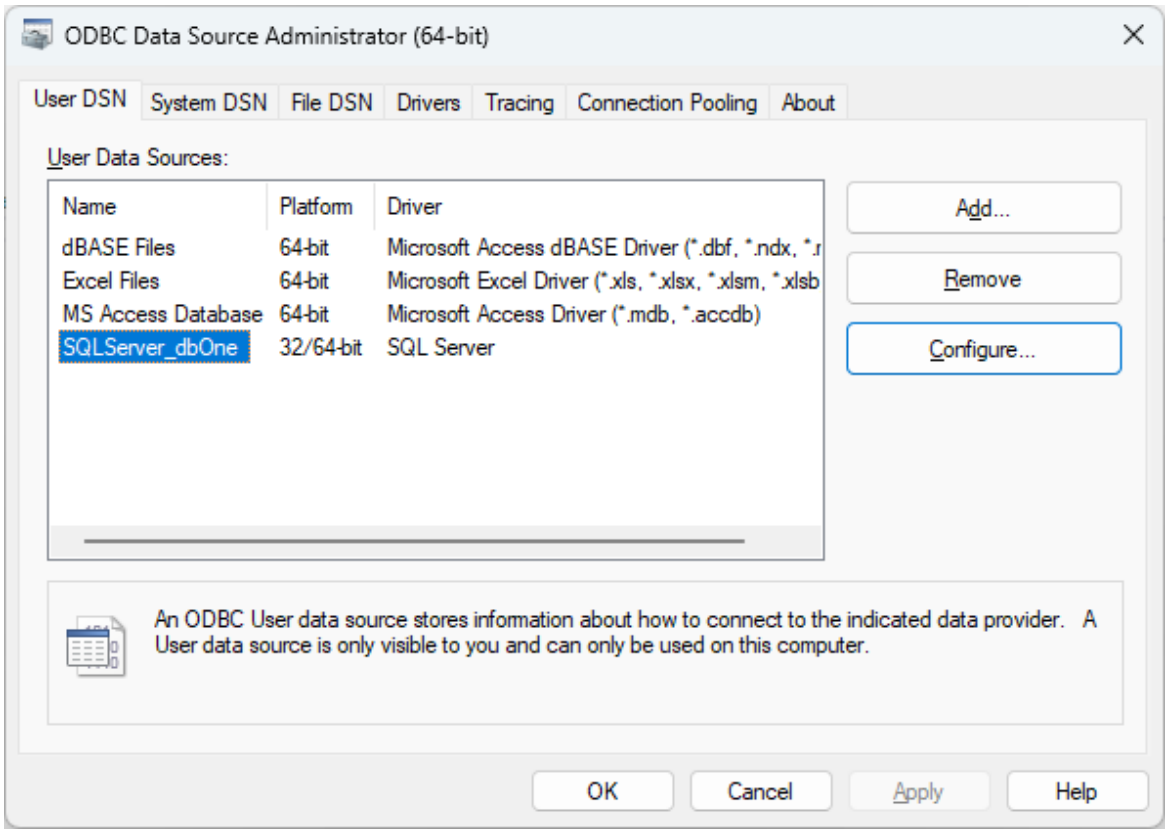
The examples in this document are for demonstration purposes, so they may not incorporate enterprise-level security measures, may use a local workstation deployment that is not amenable to sharing data, and use a simplified connection string. For simplicity, some examples do not include all APL64 executable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

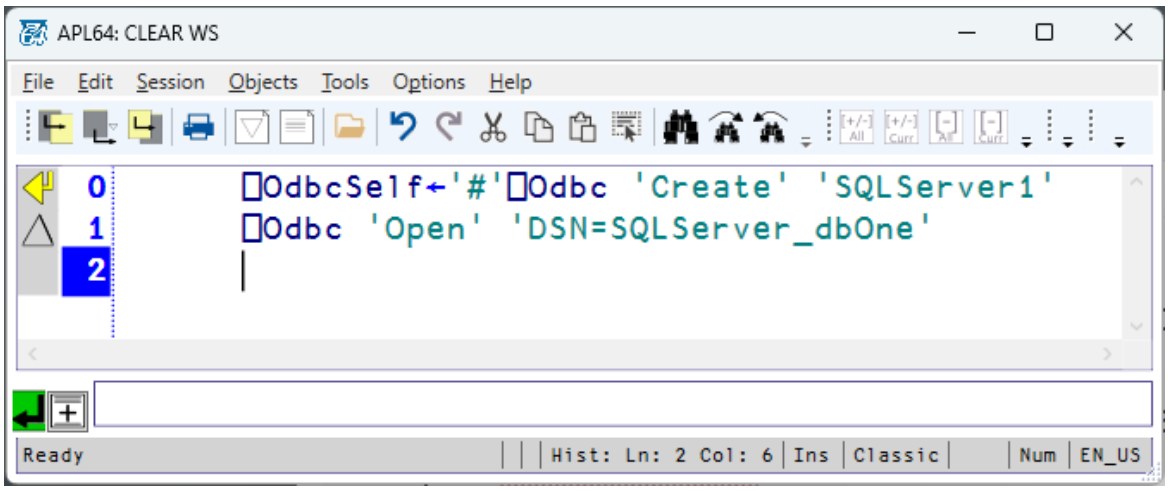
Some examples use an explicit connection string, and other examples use an ODBC DSN. In addition to creating the DSN for a database, it is also necessary to install the appropriate ODBC driver for the database.











⎕ODBC Actions

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

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

The name of an ⎕ODBC instance can be specified by:

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

⎕ODBC action names are not case sensitive. ⎕ODBC instance names are case sensitive

BeginTransaction

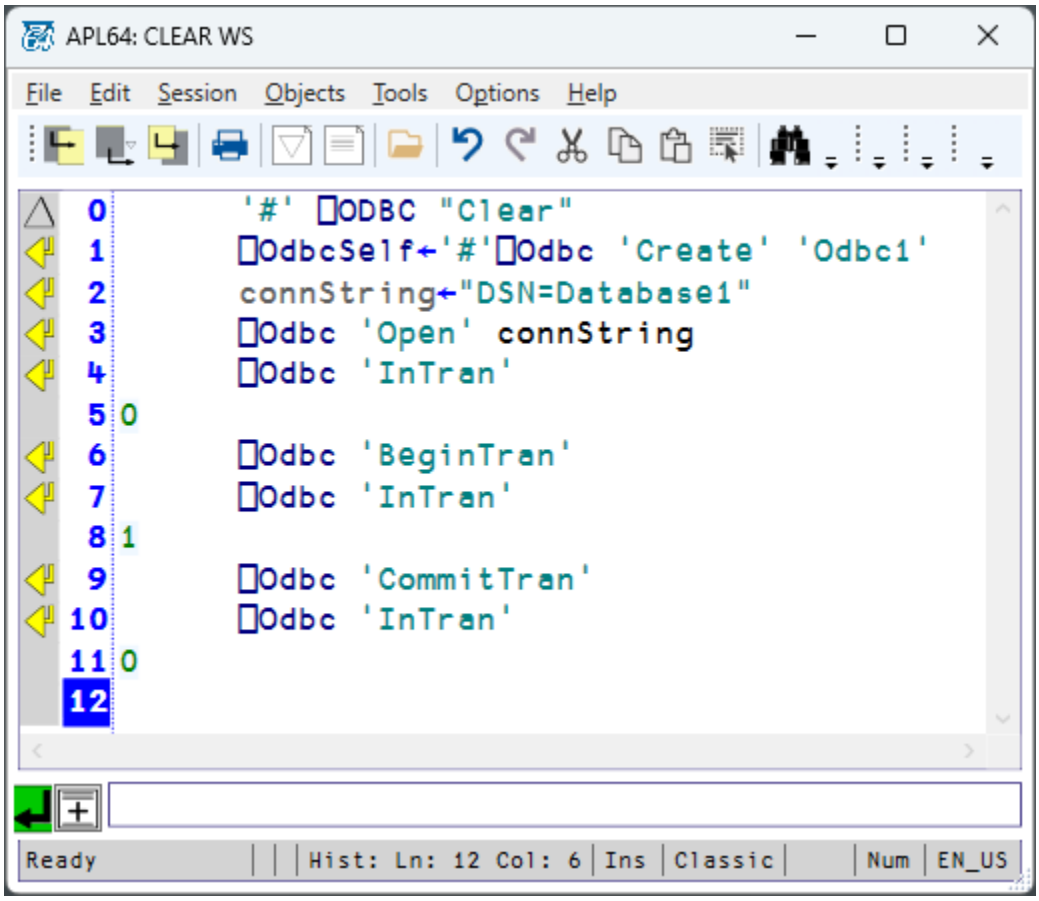
Synonym: BeginTran

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

Syntax: ⎕Odbc 'BeginTran'

ODBC transactions are used to combine multiple ODBC statements, possibly affecting multiple tables in an ODBC database, so that they are either all performed or none are performed.

```
'#' ⎕ODBC "Clear"
⎕OdbcSelf←'#'⎕Odbc 'Create' 'Odbc1'
connString←"DSN=Database1"
⎕Odbc 'Open' connString
⎕Odbc 'InTran'
⎕Odbc 'BeginTran'
⎕Odbc 'InTran'
⎕Odbc 'CommitTran'
⎕Odbc 'InTran'
```

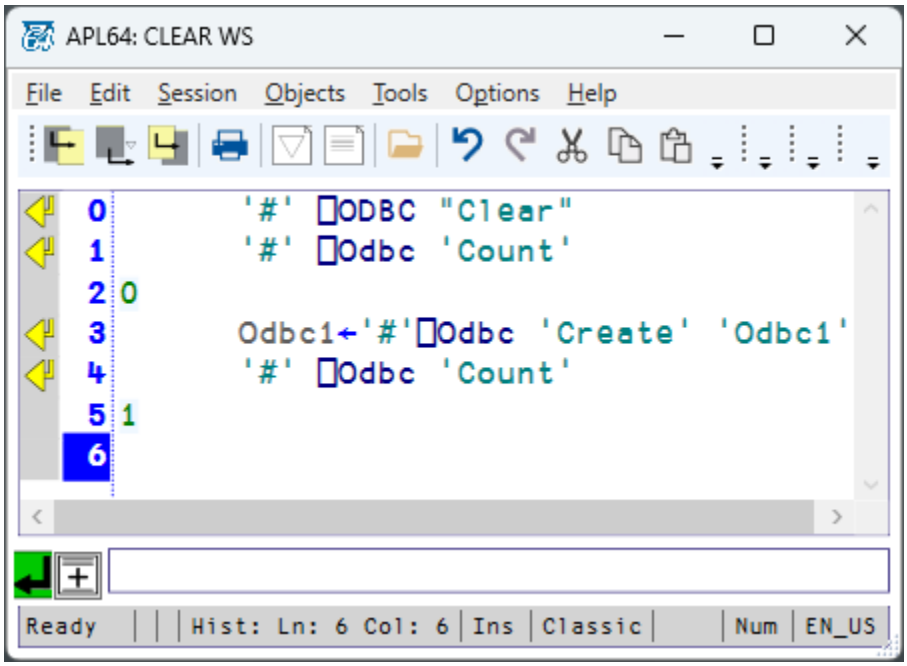


Clear

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

Syntax: '#' ⎕ODBC "Clear"

```
'#' ⎕ODBC "Clear"
'#' ⎕Odbc 'Count'
Odbc1←'#'⎕Odbc 'Create' 'Odbc1'
'#' ⎕Odbc 'Count'
```

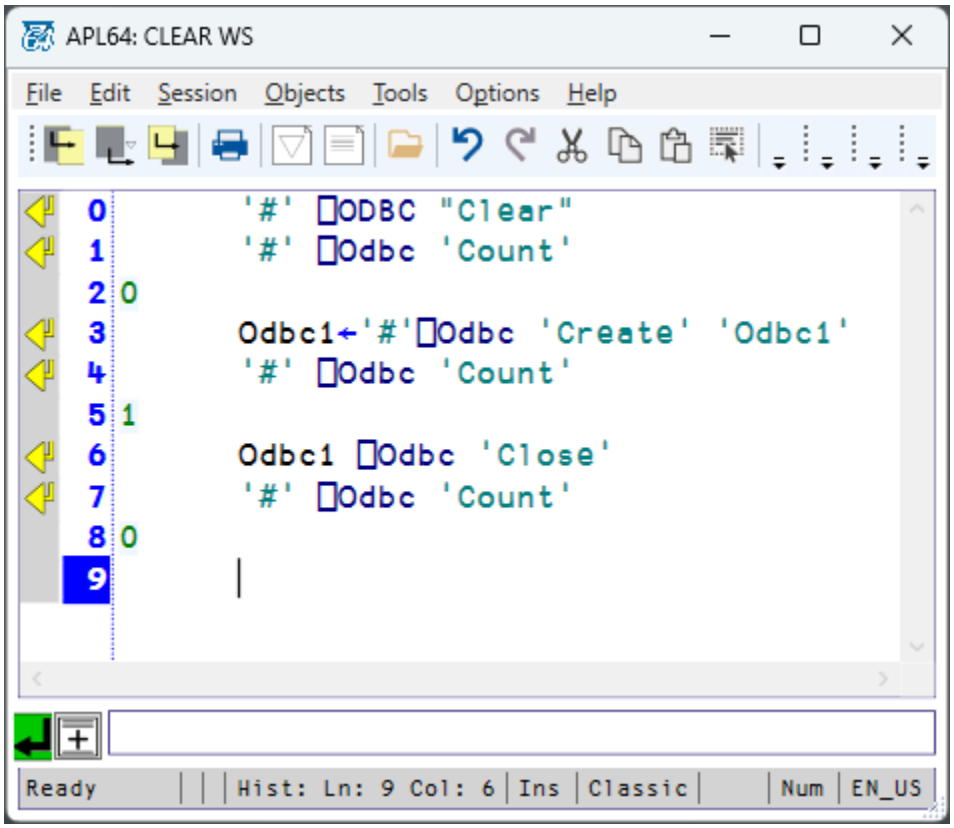


Close

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

Syntax: [instanceName] ⎕Odbc 'Close'

```
'#' ⎕ODBC "Clear"
'#' ⎕Odbc 'Count'
Odbc1←'#'⎕Odbc 'Create' 'Odbc1'
'#' ⎕Odbc 'Count'
Odbc1 ⎕Odbc 'Close'
'#' ⎕Odbc 'Count'
```



CommitTransaction

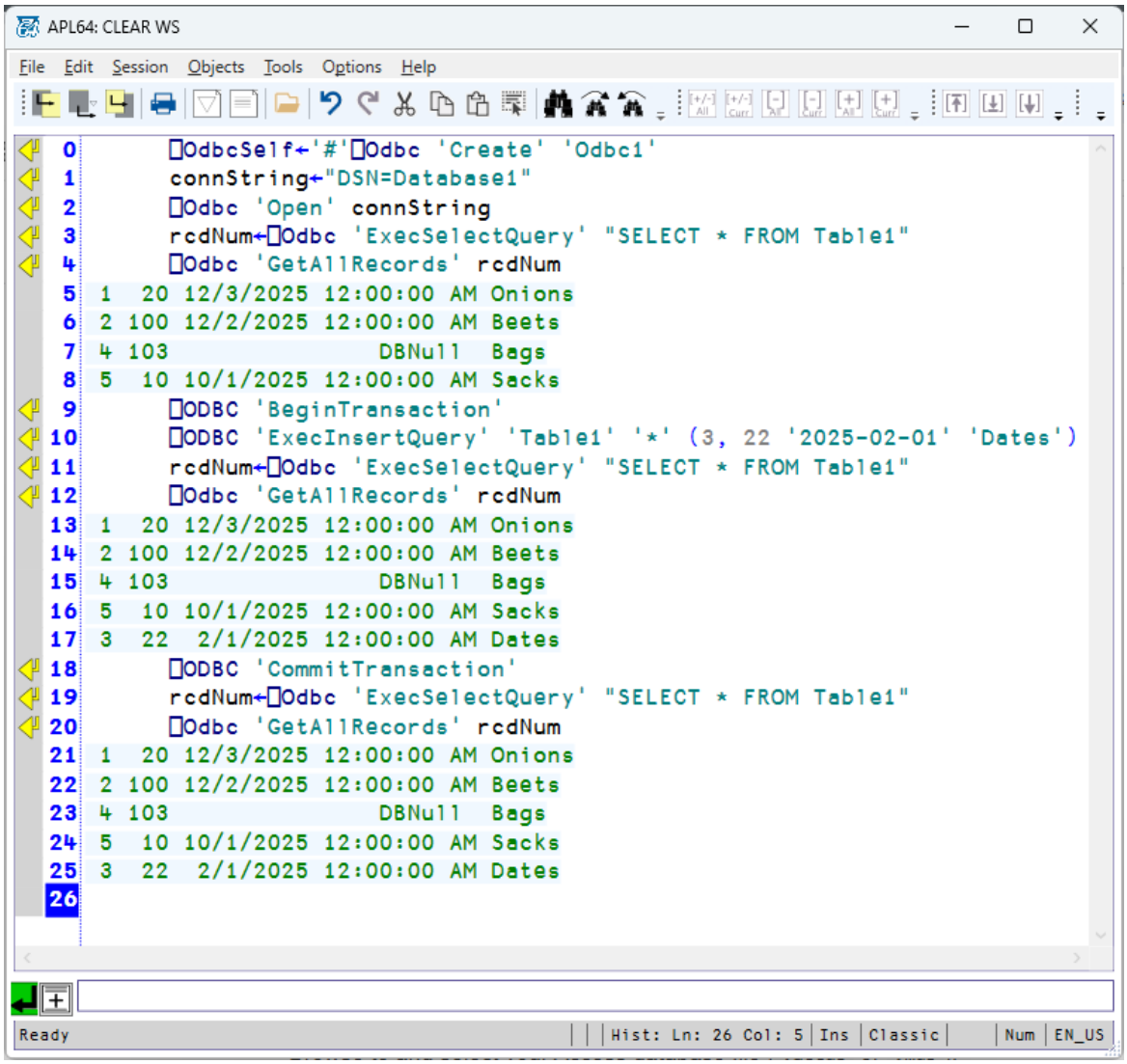
Synonym: CommitTran

This action applies to an ⎕ODBC instance. The CommitTransaction action causes the effects of the current transaction to be applied to the connected Microsoft ODBC 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 ODBC statements in the transaction are pending effects in the Odbc database. The CommitTransaction action has no result.

Syntax: [instanceName] ⎕ODBC 'CommitTransaction'

```
⎕OdbcSelf←'#'⎕Odbc 'Create' 'Odbc1'
connString←"DSN=Database1"
⎕Odbc 'Open' connString
rcdNum←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdNum
⎕ODBC 'BeginTransaction'
⎕ODBC 'ExecInsertQuery' 'Table1' '*' (3, 22 '2025-02-01' 'Dates')
rcdNum←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdNum
⎕ODBC 'CommitTransaction'
rcdNum←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
```

```
⎕Odbc 'GetAllRecords' rcdNum
```



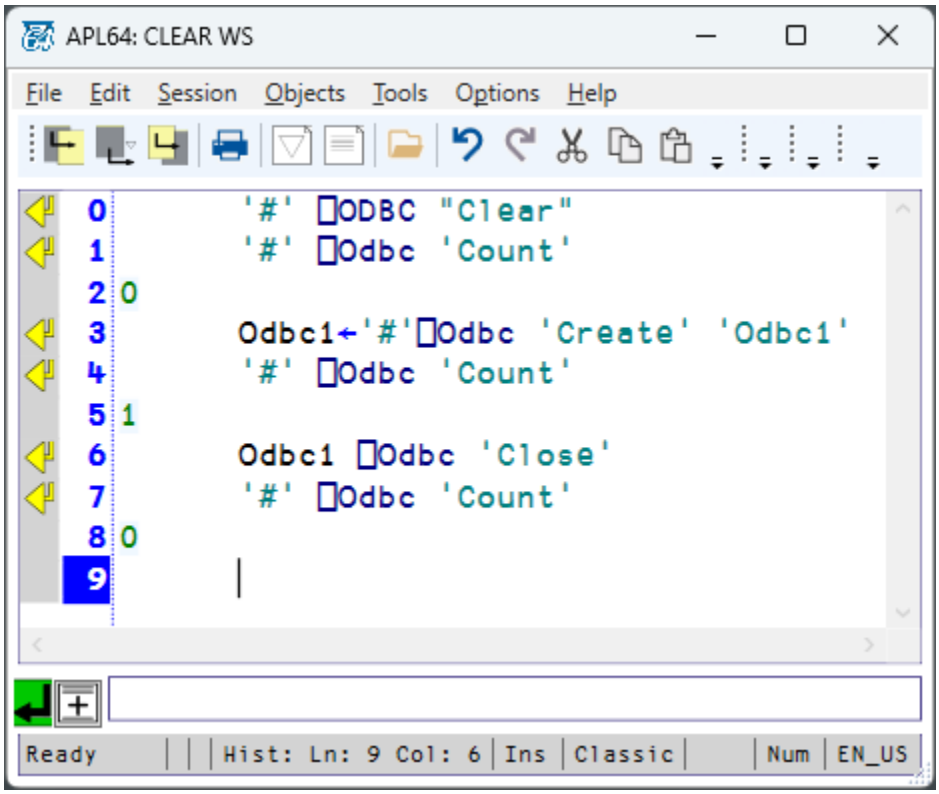
Count

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

Syntax: '#' ⎕Odbc 'Count'

```
'#' ⎕ODBC "Clear"
'#' ⎕Odbc 'Count'
Odbc1←'#'⎕Odbc 'Create' 'Odbc1'
'#' ⎕Odbc 'Count'
Odbc1 ⎕Odbc 'Close'
```

```
'#' ⎕Odbc 'Count'
```



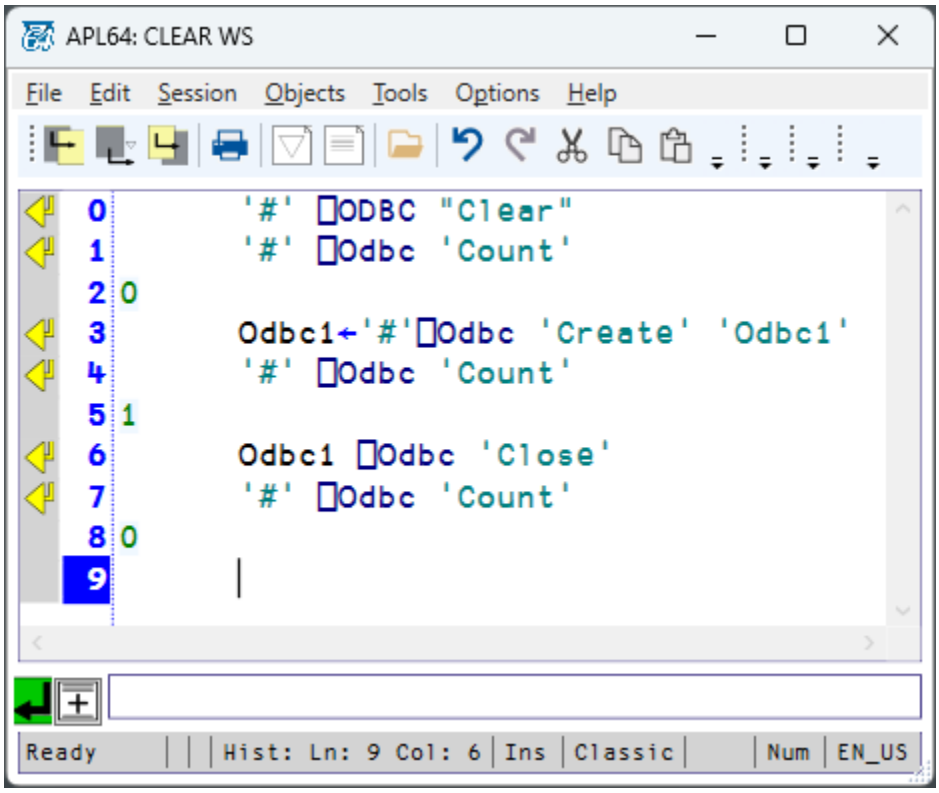
Create

This action is performed on the ⎕ODBC object. The Create action will create an ⎕ODBC instance with a user-provided name in the right argument. Multiple ⎕ODBC instances are possible in the same APL64 instance, so that multiple ODBC 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 ODBC database is generally used once in an APL64 instance. The Create action does not open a connection to an ODBC database, use the Open action for that purpose. The result of a successful 'Create' action is a text vector containing the ⎕ODBC instance name.

Syntax: instanceName←'#'⎕Odbc 'Create' instanceName

```
'#' ⎕ODBC "Clear"
'#' ⎕Odbc 'Count'
Odbc1←'#'⎕Odbc 'Create' 'Odbc1'
'#' ⎕Odbc 'Count'
Odbc1 ⎕Odbc 'Close'
'#' ⎕Odbc 'Count'
```

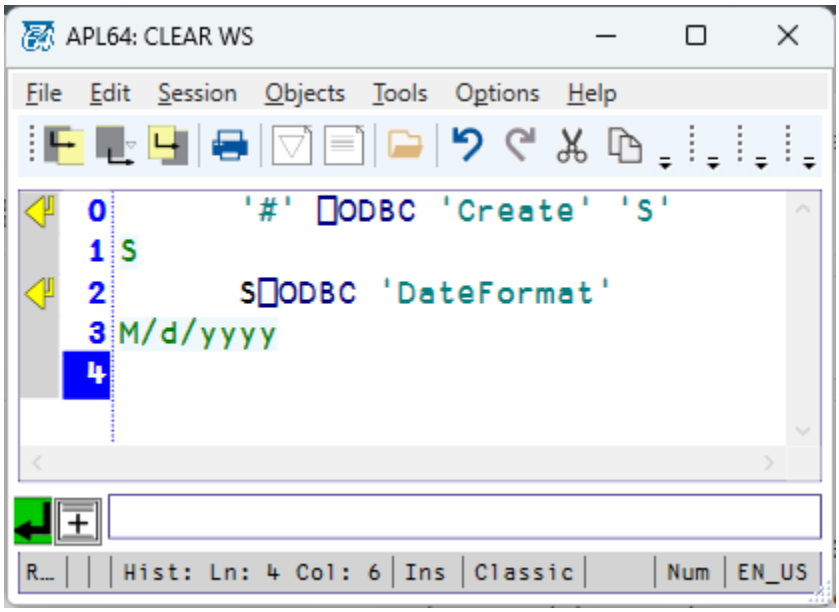


DateFormat

This action returns the current .Net Short Date Pattern text applicable to the specified ⎕ODBC instance.

Syntax: [instanceName] ⎕ODBC 'DateFormat'

```
'#' ⎕ODBC 'Create' 'S'
S⎕ODBC 'DateFormat'
```

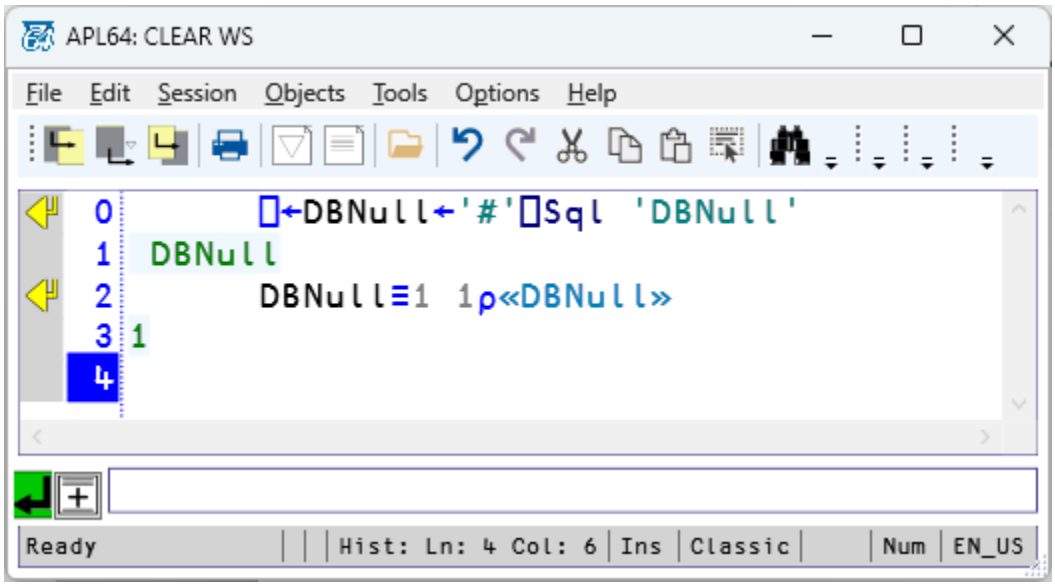


DBNull

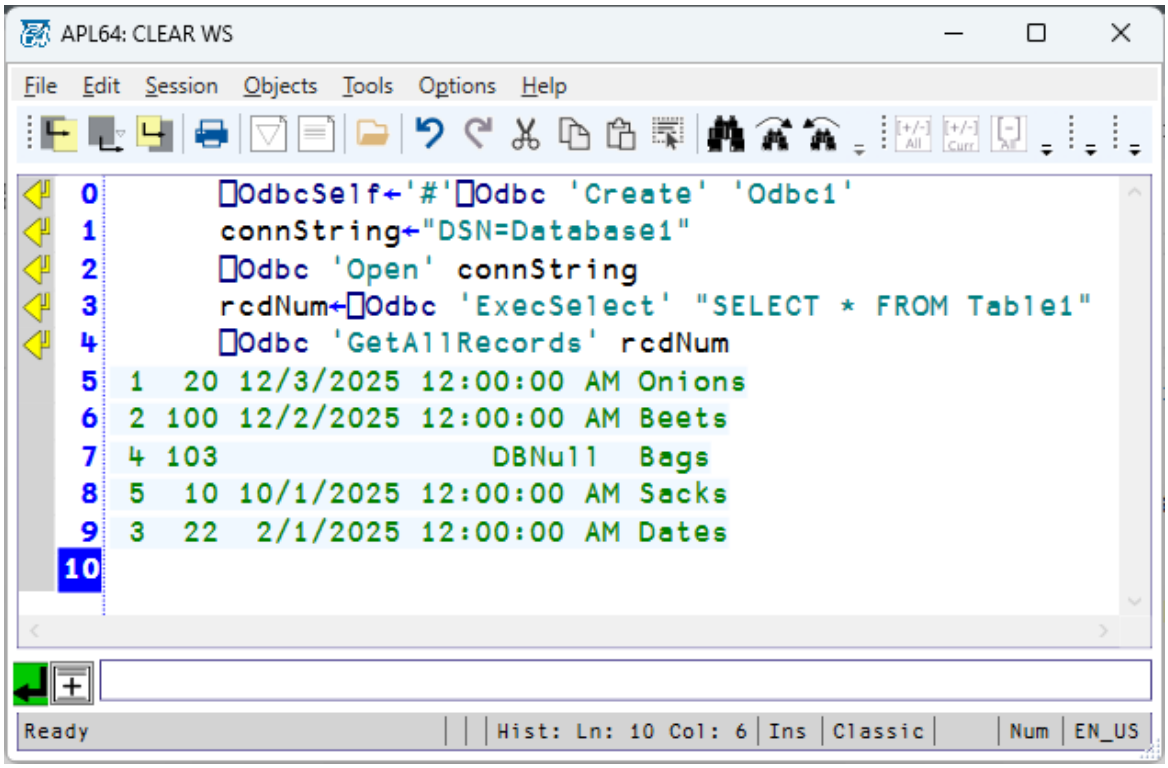
This action returns a value which may be used to specify null values for the ExecInsertQuery and ExecStoredProc ⎕ODBC instance actions. This value will also be used as the APL64 value associated with a DBNull value returned from a GetRecord, GetAllRecords, or ExecStoredProc action.

Syntax: DBNull←'#'⎕ODBC 'DBNull'

```
⎕←DBNull←'#'⎕ODBC 'DBNull'
DBNull≡1 1⍴«DBNull»
```



```
⎕OdbcSelf←'#'⎕Odbc 'Create' 'Odbc1'
connString←"DSN=Database1"
⎕Odbc 'Open' connString
rcdNum←⎕Odbc 'ExecSelect' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdNum
```

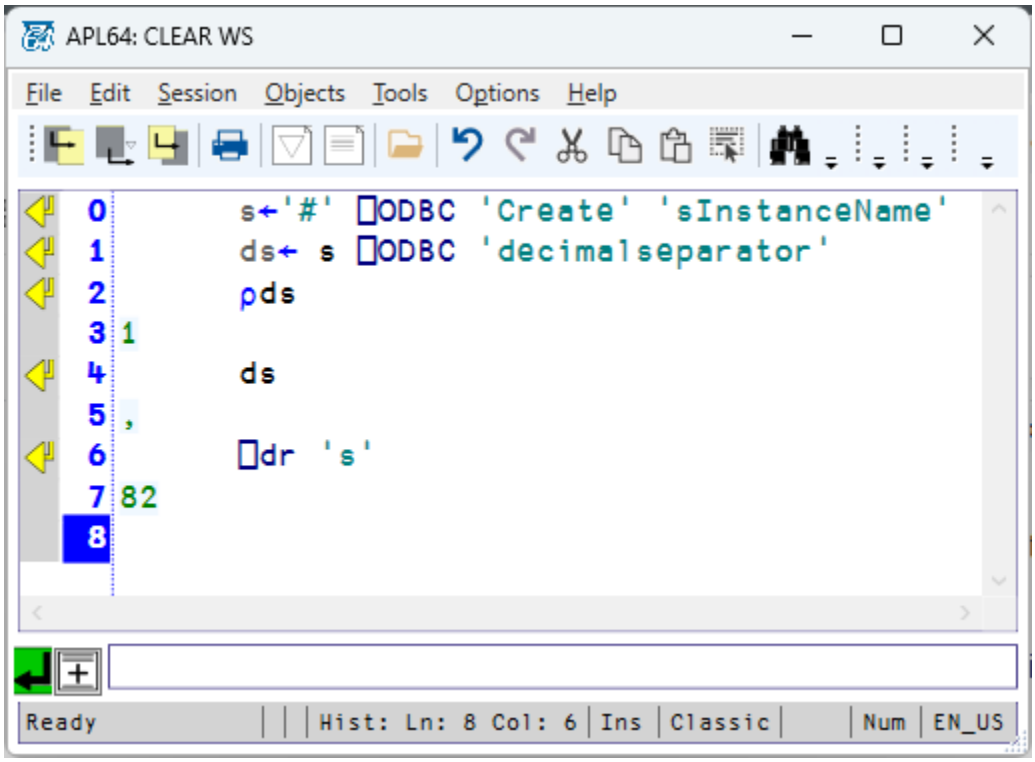


DecimalSeparator

This action returns the current .Net Number Decimal Separator text applicable to the specified ⎕ODBC instance.

Syntax: value← [instanceName] ⎕ODBC 'decimalseparator'

```
s←'#' ⎕ODBC 'Create' 'sInstanceName'
ds← s ⎕ODBC 'decimalseparator'
⍴ds
ds
⎕dr 's'
```

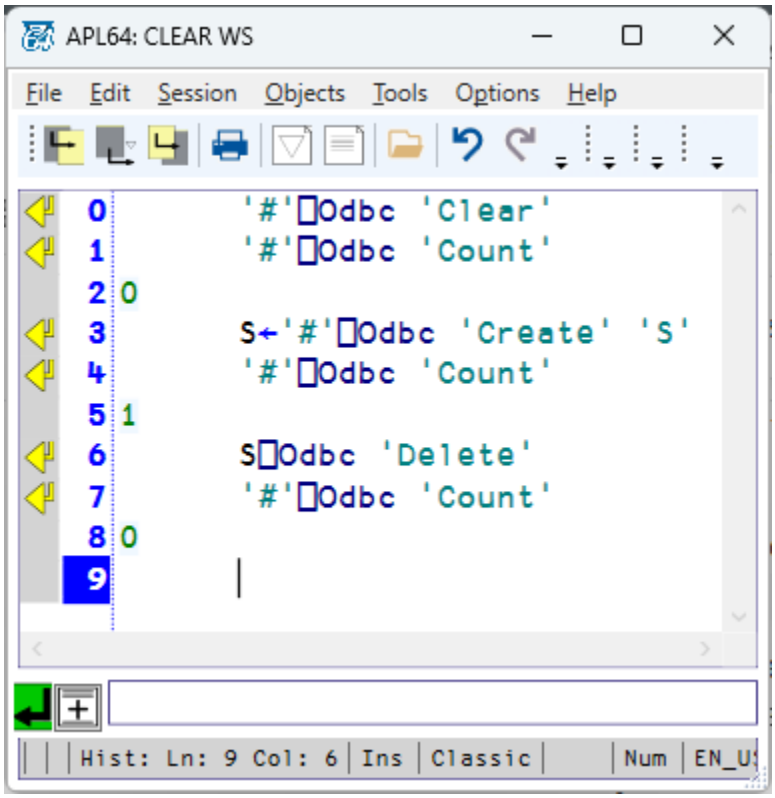


Delete

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

Syntax: [instanceName] ⎕Odbc 'Delete'

```
'#'⎕Odbc 'Clear'
'#'⎕Odbc 'Count'
S←'#'⎕Odbc 'Create' 'S'
'#'⎕Odbc 'Count'
S⎕Odbc 'Delete'
'#'⎕Odbc 'Count'
```



Exec

This action applies to an ⎕ODBC instance. The ⎕ODBC Exec action returns the number of database rows, if any, affected by the executed ODBC statement provided by the right argument. For certain actions, such as creating a table, the result of the Exec action can be negative. The ⎕ODBC Exec action cannot be used to return results of a Select ODBC statement. The ⎕ODBC action can perform any actions permitted on the database, including add/delete tables, etc.

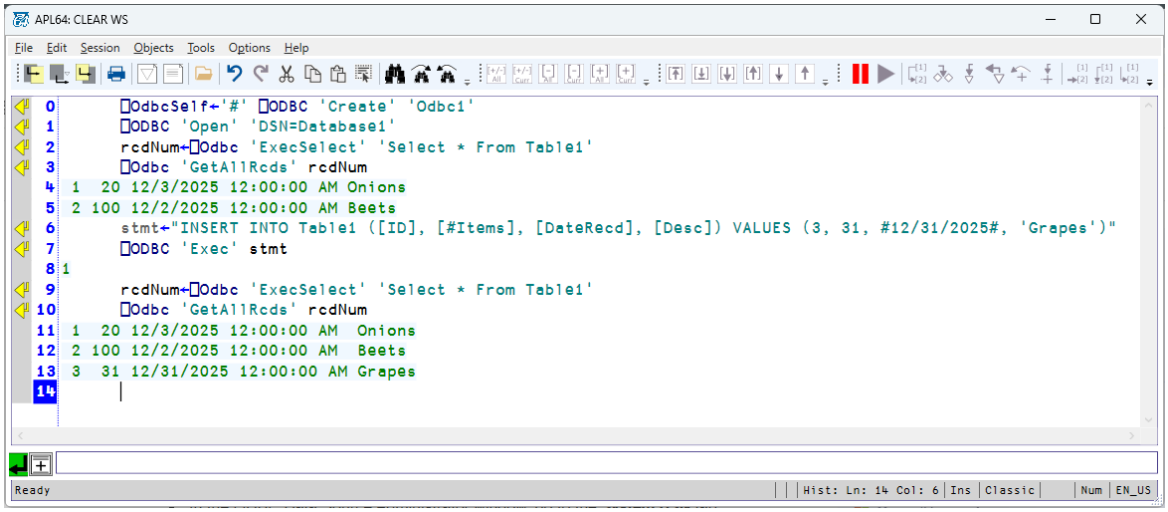
Syntax: result←⎕ODBC 'Exec' stmt [subValue1] [subValue2]...

APL64 values may be substituted into the Exec statement. The substitution location in the Exec statement 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 statement.

Example: Insert action

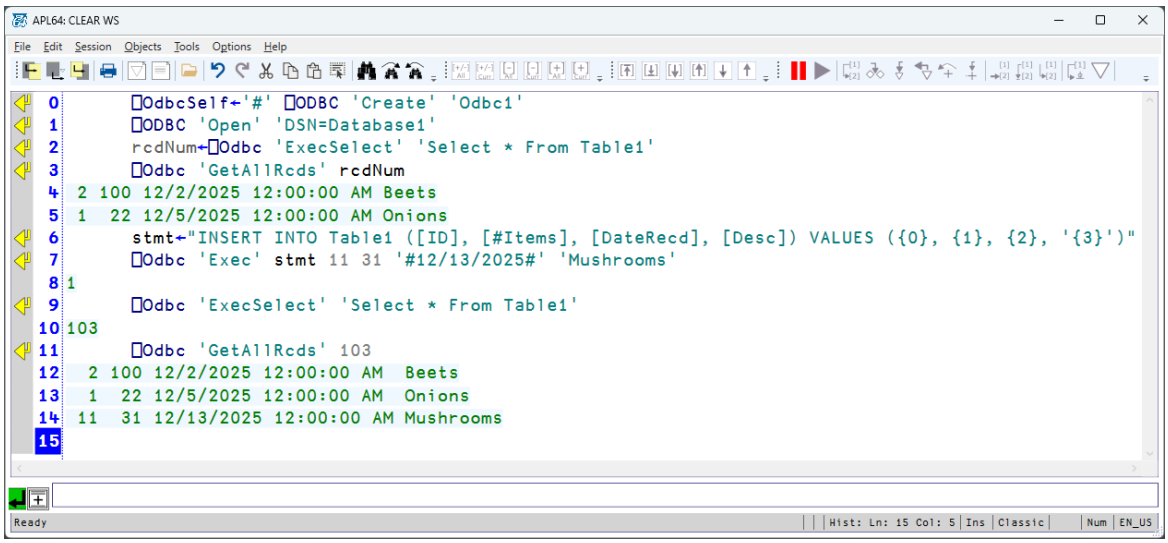
```
⎕OdbcSelf←'#' ⎕ODBC 'Create' 'Odbc1'
⎕ODBC 'Open' 'DSN=Database1'
rcdNum←⎕Odbc 'ExecSelect' 'Select * From Table1'
⎕Odbc 'GetAllRcds' rcdNum
stmt←"INSERT INTO Table1 ([ID], [#Items], [DateRecd], [Desc]) VALUES (3, 31, #12/31/2025#,
'Grapes')"
```

```
⎕ODBC 'Exec' stmt
rcdNum←⎕Odbc 'ExecSelect' 'Select * From Table1'
⎕Odbc 'GetAllRcds' rcdNum
```



Example: Insert action with value substitution

```
⎕OdbcSelf←'#' ⎕ODBC 'Create' 'Odbc1'
⎕ODBC 'Open' 'DSN=Database1'
rcdNum←⎕Odbc 'ExecSelect' 'Select * From Table1'
⎕Odbc 'GetAllRcds' rcdNum
stmt←"INSERT INTO Table1 ([ID], [#Items], [DateRecd], [Desc]) VALUES ({0}, {1}, {2}, '{3}'"
⎕ODBC 'Exec' stmt 11 31 '#12/13/2025#' 'Mushrooms'
⎕Odbc 'ExecSelect' 'Select * From Table1'
⎕Odbc 'GetAllRcds' 103
```



ExecDeleteQuery

Synonym: ExecDelete

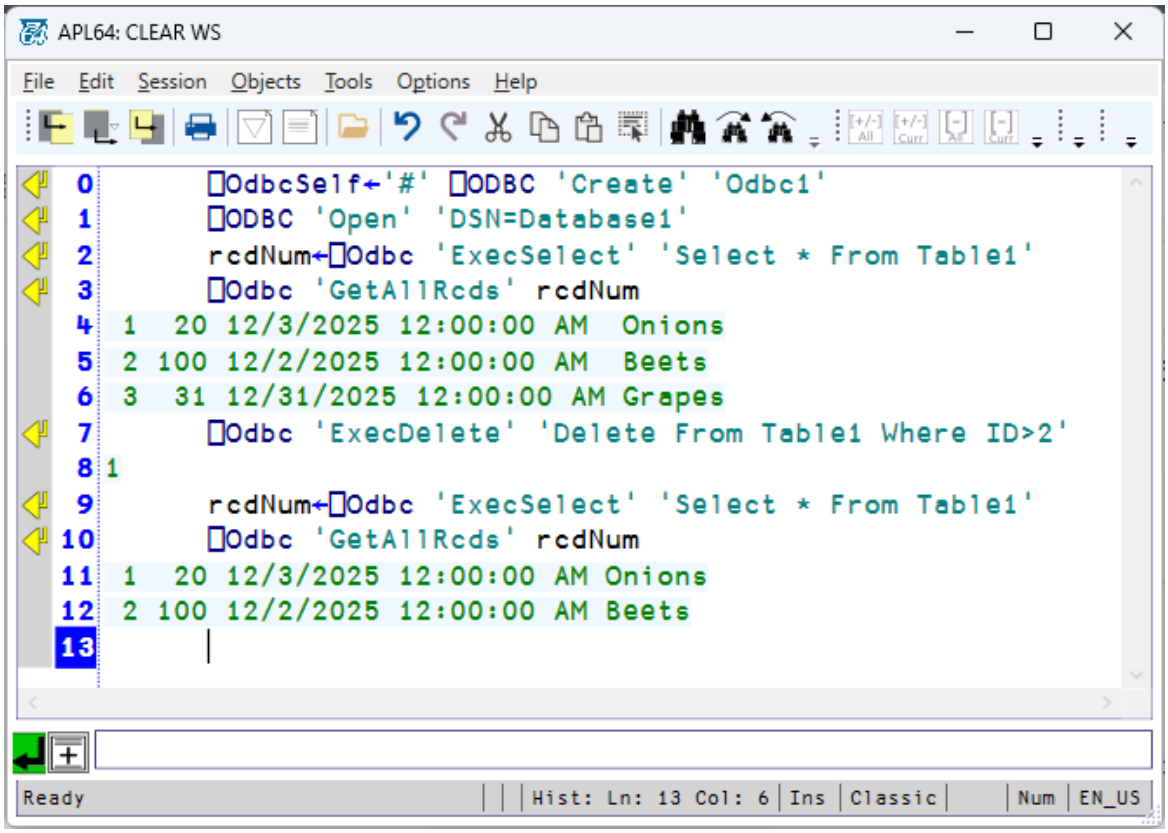
This action applies to an ⎕ODBC instance. The right argument is the text of the ODBC Delete statement. The result of the ⎕ODBC ExecDeleteQuery is the number of records affected, if any. This action is used for delete actions which are transiently created at runtime.

Syntax: result← [instanceName] ⎕Odbc 'ExecDelete' stmt [subValue1] [subValue2]…

APL64 values may be substituted into the ExecDelete statement. The substitution location in the Exec statement 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 ExecDelete statement.

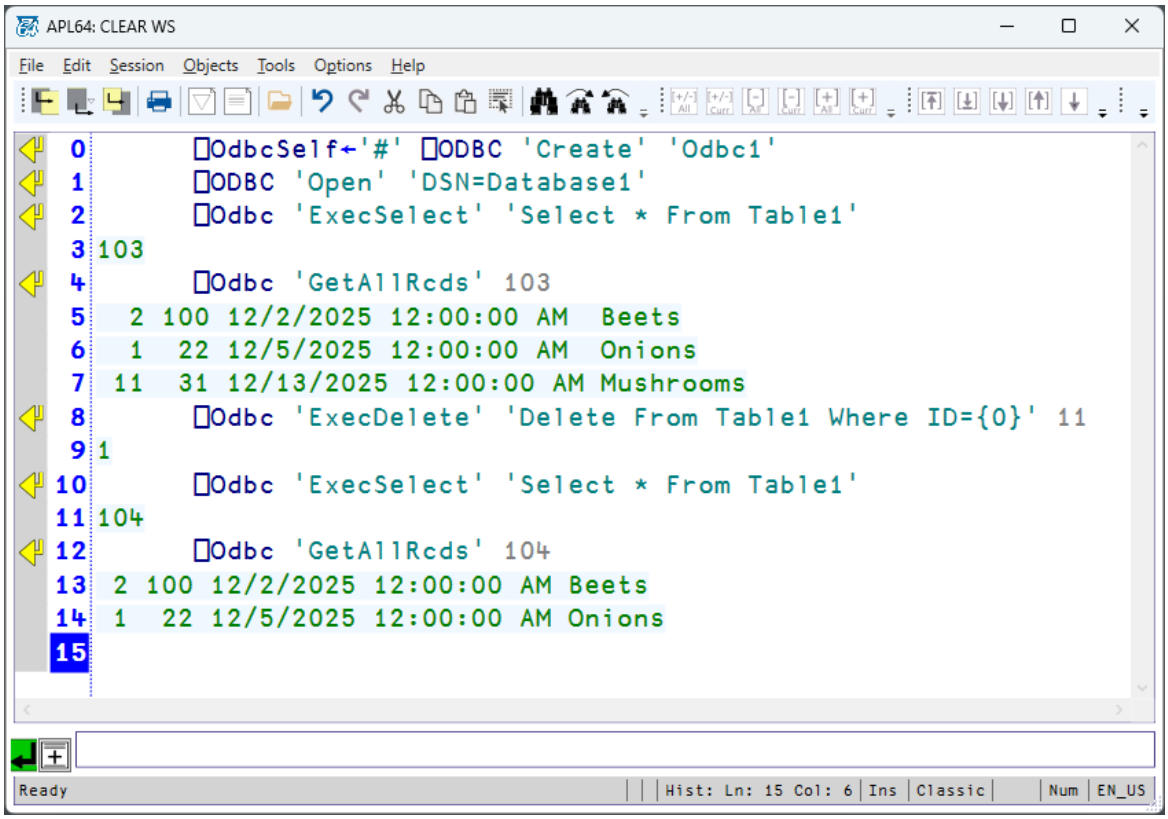
Example: ExecDelete

```
⎕OdbcSelf←'#' ⎕ODBC 'Create' 'Odbc1'
⎕ODBC 'Open' 'DSN=Database1'
rcdNum←⎕Odbc 'ExecSelect' 'Select * From Table1'
⎕Odbc 'GetAllRcds' rcdNum
⎕Odbc 'ExecDelete' 'Delete From Table1 Where ID>2'
rcdNum←⎕Odbc 'ExecSelect' 'Select * From Table1'
⎕Odbc 'GetAllRcds' rcdNum
```



Example: ExecDelete with Value Substitution

```
⎕OdbcSelf←'#' ⎕ODBC 'Create' 'Odbc1'
⎕ODBC 'Open' 'DSN=Database1'
⎕Odbc 'ExecSelect' 'Select * From Table1'
⎕Odbc 'GetAllRcds' 103
⎕Odbc 'ExecDelete' 'Delete From Table1 Where ID={0}' 11
⎕Odbc 'ExecSelect' 'Select * From Table1'
⎕Odbc 'GetAllRcds' 104
```



ExecInsertQuery

Synonym: ExecInsert

This action will insert APL64-compatible data into a target table of the database. This action applies to an ⎕ODBC instance. This action is used for select queries which are transiently created at run time.

Syntax: ⎕Odbc 'ExecInsert' tableName 'Field1,…' aplMatrix

Right arguments:

Odbc table name: The name of the database table into which records will be inserted

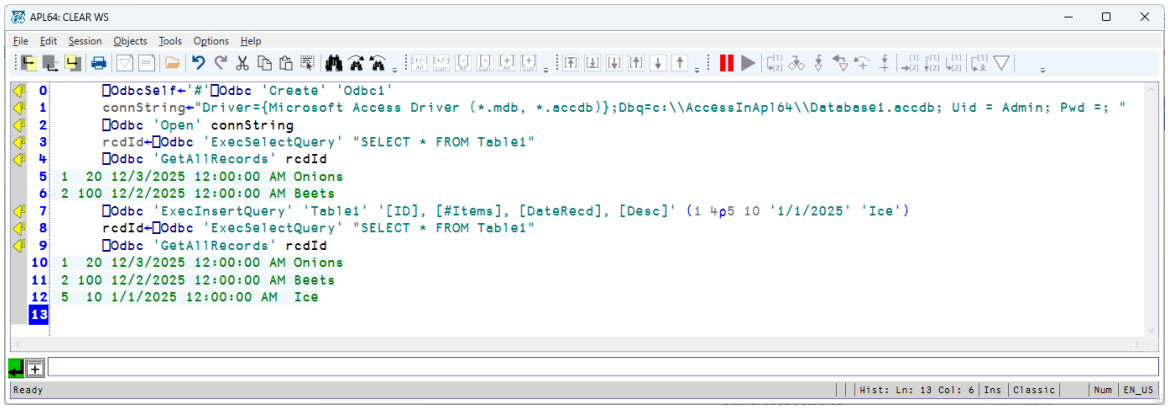
Odbc column names: Comma delimited text with selected Odbc table column names, or '*' to indicate all Odbc table column names.

APL64 data: A matrix containing the APL64 values to be inserted. One row for each record to be inserted, with one column for each specified column name.

Auto-generated column values for a record should not be inserted using this action. The order of the Odbc column names and APL64 data matrix columns must be conformable.

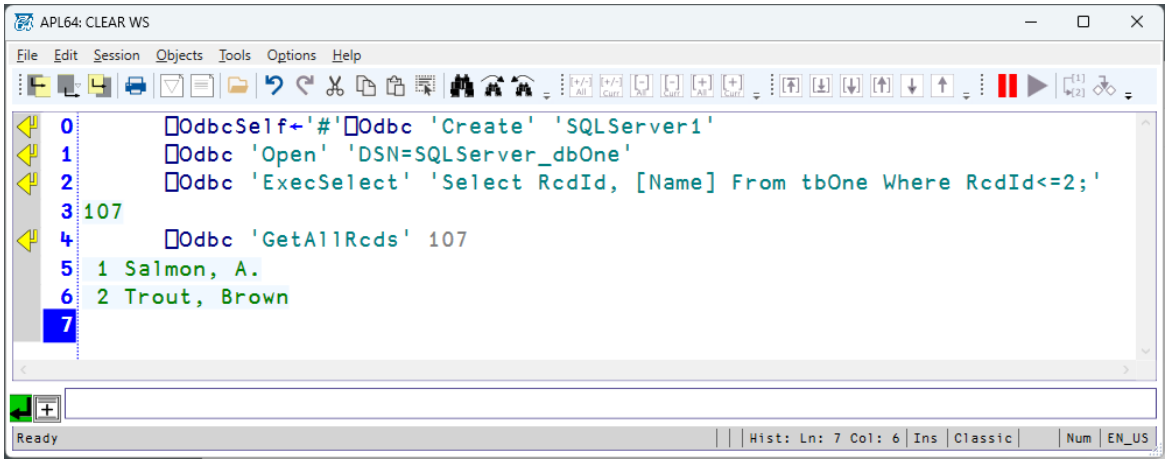
Example: Specify all Column Names in the Table

```
⎕OdbcSelf←'#'⎕Odbc 'Create' 'Odbc1'
connString←"Driver={Microsoft Access Driver(
*.mdb,*.accdb)};Dbq=c:\\AccessInApl64\\Database1.accdb; Uid = Admin; Pwd =; "
⎕Odbc 'Open' connString
rcdId←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdId
 12/3/2025 12:00:00 AM Onions
 12/2/2025 12:00:00 AM Beets
⎕Odbc 'ExecInsertQuery' 'Table1' '[ID], [#Items], [DateRecd], [Desc]' (1 4⍴5 10 '1/1/2025' 'Ice')
rcdId←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdId
```



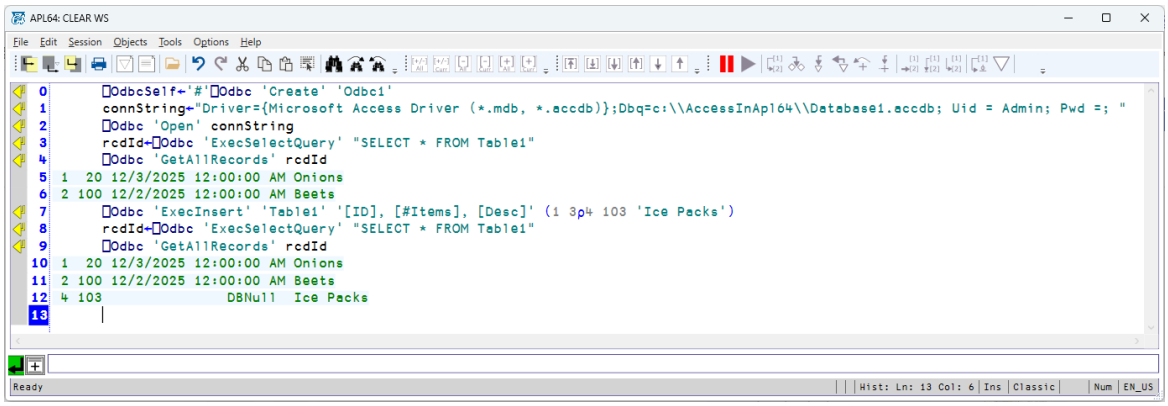
Example: Specify some, but not all, Column Names in the Table

```
⎕OdbcSelf←'#'⎕Odbc 'Create' 'SQLServer1'
⎕Odbc 'Open' 'DSN=SQLServer_dbOne'
⎕Odbc 'ExecSelect' 'Select RcdId, [Name] From tbOne Where RcdId<=2;'
⎕Odbc 'GetAllRcds' 107
```



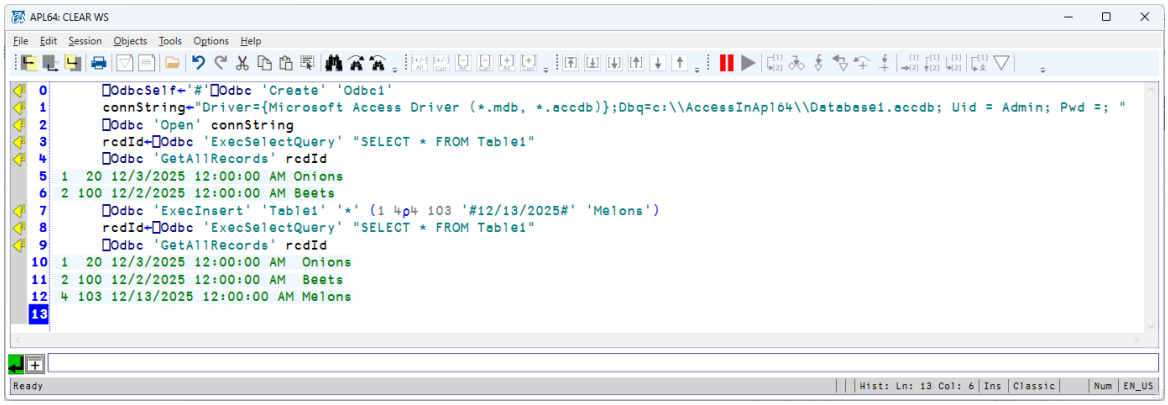

```
⎕OdbcSelf←'#'⎕Odbc 'Create' 'Odbc1'
```

```
connString←"Driver={Microsoft Access Driver (
*.mdb, *.accdb)};Dbq=c:\\AccessInApl64\\Database1.accdb; Uid = Admin; Pwd =; "
⎕Odbc 'Open' connString
rcdId←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdId
⎕Odbc 'ExecInsert' 'Table1' '[ID], [#Items], [Desc]' (1 3⍴4 103 'Ice Packs')
rcdId←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdId
```



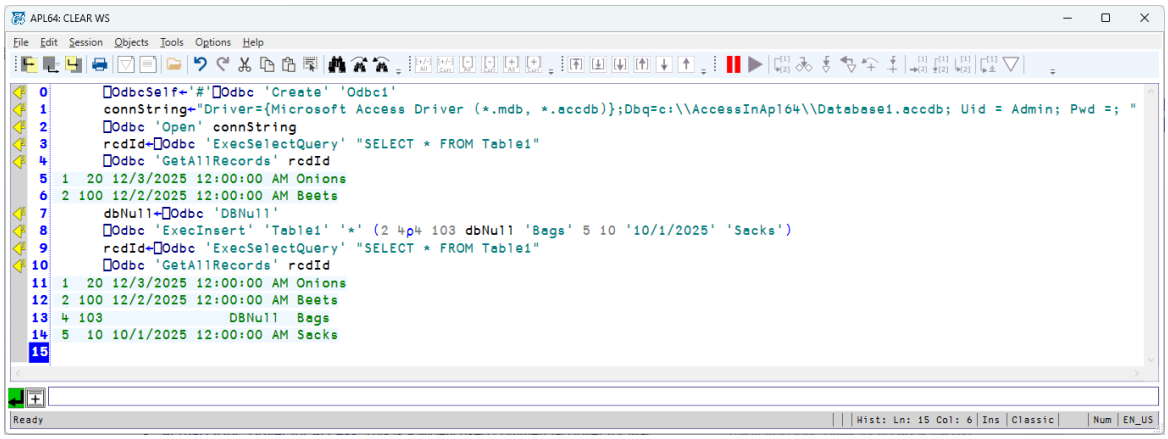
Example: Specify all table columns using *

```
⎕OdbcSelf←'#'⎕Odbc 'Create' 'Odbc1'
connString←"Driver={Microsoft Access Driver (*.mdb,
*.accdb)};Dbq=c:\\AccessInApl64\\Database1.accdb; Uid = Admin; Pwd =; "
⎕Odbc 'Open' connString
rcdId←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdId
 12/3/2025 12:00:00 AM Onions
 12/2/2025 12:00:00 AM Beets
⎕Odbc 'ExecInsert' 'Table1' '*' (1 4⍴4 103 '#12/13/2025#' 'Melons')
rcdId←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdId
```



Example: Specify all columns using *, some values as DBNull, multiple records returned

```
⎕OdbcSelf←'#'⎕Odbc 'Create' 'Odbc1'
connString←"Driver={Microsoft Access Driver (*.mdb, *.accdb)};Dbq=c:\\AccessInApl64\\Database1.accdb; Uid = Admin; Pwd =; "
⎕Odbc 'Open' connString
rcdId←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdId
 12/3/2025 12:00:00 AM Onions
 12/2/2025 12:00:00 AM Beets
dbNull←⎕Odbc 'DBNull'
⎕Odbc 'ExecInsert' 'Table1' '*' (2 4⍴4 103 dbNull 'Bags' 5 10 '10/1/2025' 'Sacks')
rcdId←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdId
```



ExecSelectQuery

Synonym: ExecSelect

The ExecSelectQuery action applies to an ⎕ODBC instance. The action argument provides the ODBC Select statement. Depending on the Odbc 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: result←⎕Odbc 'ExecSelect' stmt [subValue1] [subValue2]…

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

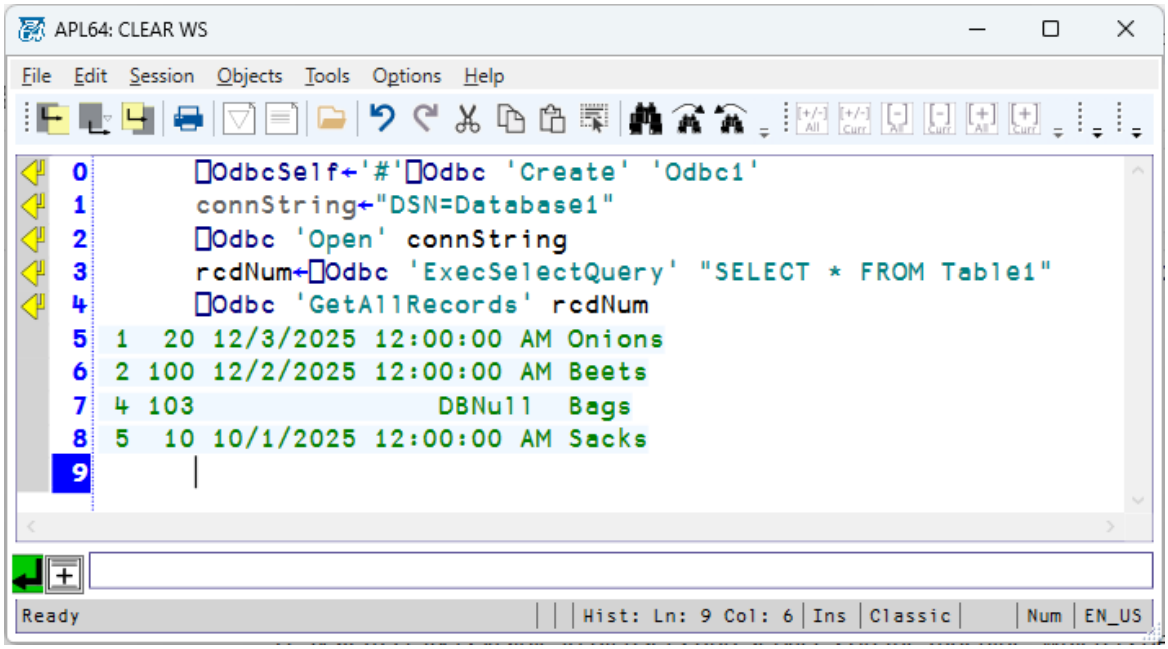
The ODBC-format query text argument to the ⎕ODBC ExecSelectQuery method, should yield only one record set. A query of the form 'query1;…;queryN', is not supported in APL64 and must be separated into individual queries. As an alternative to separate queries, a stored procedure can

be created and executed using the ⎕ODBC ExecStoredProc method, or several queries can be included in an ODBC transaction.

APL64 values may be substituted into the ExecSelect statement. The substitution location in the Exec statement 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 ExecSelect statement.

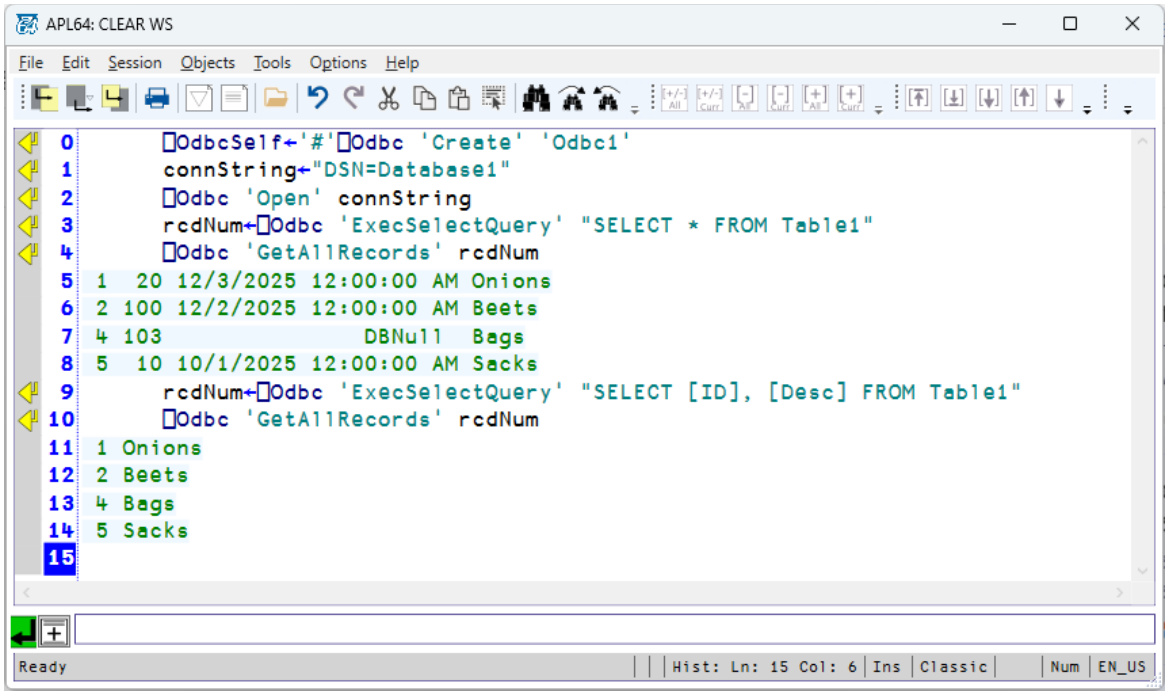
Example: Select all columns and all records

```
⎕OdbcSelf←'#'⎕Odbc 'Create' 'Odbc1'
connString←"DSN=Database1"
⎕Odbc 'Open' connString
rcdNum←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdNum
```



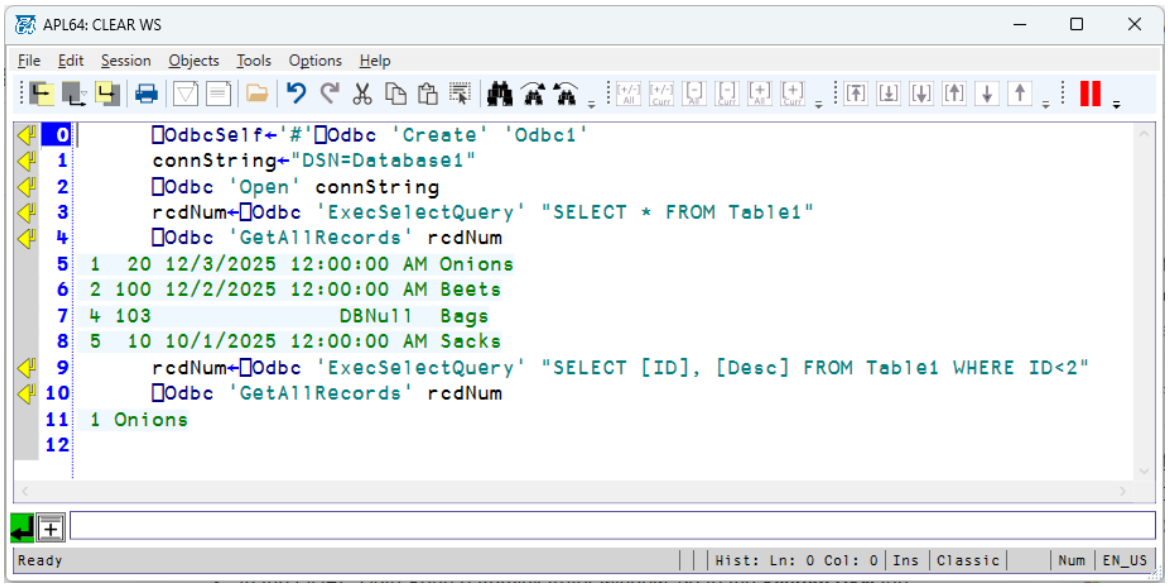
Example: Select some columns and all records

```
⎕OdbcSelf←'#'⎕Odbc 'Create' 'Odbc1'
connString←"DSN=Database1"
⎕Odbc 'Open' connString
rcdNum←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdNum
rcdNum←⎕Odbc 'ExecSelectQuery' "SELECT [ID], [Desc] FROM Table1"
⎕Odbc 'GetAllRecords' rcdNum
```



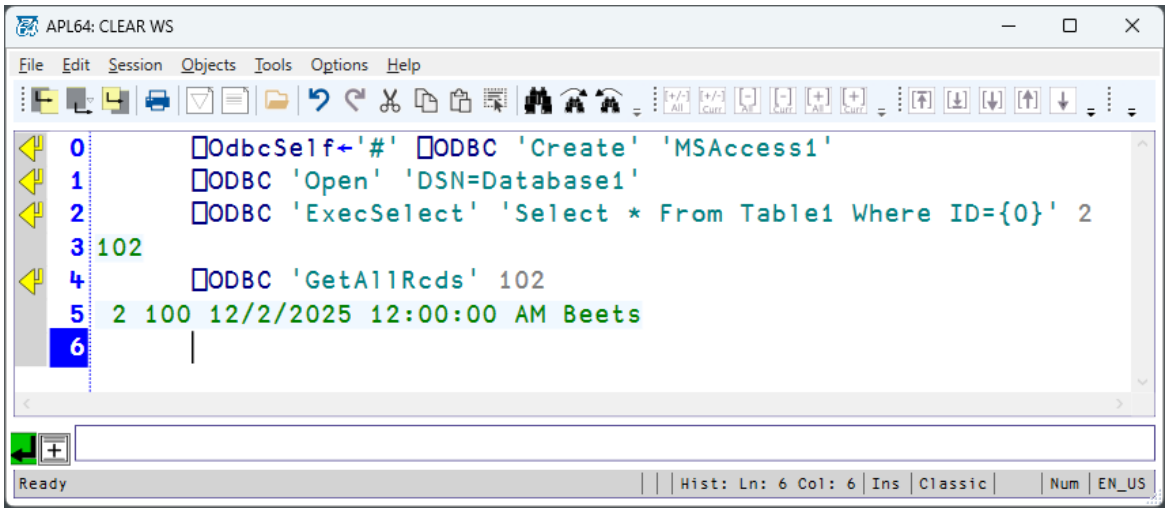
Example: Select some, but not all, records

```
⎕OdbcSelf←'#'⎕Odbc 'Create' 'Odbc1'
connString←"DSN=Database1"
⎕Odbc 'Open' connString
rcdNum←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdNum
rcdNum←⎕Odbc 'ExecSelectQuery' "SELECT [ID], [Desc] FROM Table1 WHERE ID<2"
⎕Odbc 'GetAllRecords' rcdNum
```



Example: ExecSelect with value substitution

```
⎕OdbcSelf←'#' ⎕ODBC 'Create' 'MSAccess1'
⎕ODBC 'Open' 'DSN=Database1'
⎕ODBC 'ExecSelect' 'Select * From Table1 Where ID={0}' 2
⎕ODBC 'GetAllRcds' 102
```



ExecStoredProc

This action can be used to run a stored procedure. Some database types do not support stored procedures. Before using the ExecStoredProc action, the stored procedure should be thoroughly tested within the database software GUI.

Procedure parameters are variables in the procedure code which will be provided at runtime. Parameters are optional. Incorrect parameter values will lead to exceptions which prevent processing of the ⎕ODBC ExecStoredProc Action

For Microsoft Access using the Ace ODBC driver, the stored procedure must be a 'saved query', with none or more input parameters. Output, Input/Output and ReturnValue parameters are not supported due to limitations of the Ace ODBC driver, and Access Trust Center considerations.

For other database software, support for stored procedures depends on the ODBC driver software installed. For Microsoft SQL Server input, Output, Input/Output and ReturnValue parameters are supported.

Syntax: resultMatrix ←[instanceName] ⎕ODBC 'ExecStoredProc' procName procData

procName: Text containing the stored procedure name

procData: Matrix with one row for each parameter defined in the stored procedure. The order of the rows must correspond to the order of the parameters in the stored procedure. All columns must be present in procData.

procData Columns	
Col#	**Parameter Item**
1	Parameter name, including @-prefix or […], if present in definition
2	ODBC data type
3	Direction: Input, Output, InputOutput, ReturnValue
4	Size: Integer as defined in the definition, otherwise 0
5	APL Value: For Input and InputOutput parameters, otherwise 0

The data type specified for a parameter in the stored procedure may not correspond to an Odbc data type. Not all Odbc data types can be represented in APL64. The stored procedure definition must cast parameter values to/from APL64-compatible data types.

resultMatrix: 2-column matrix with one row for each result, out and inout parameter:

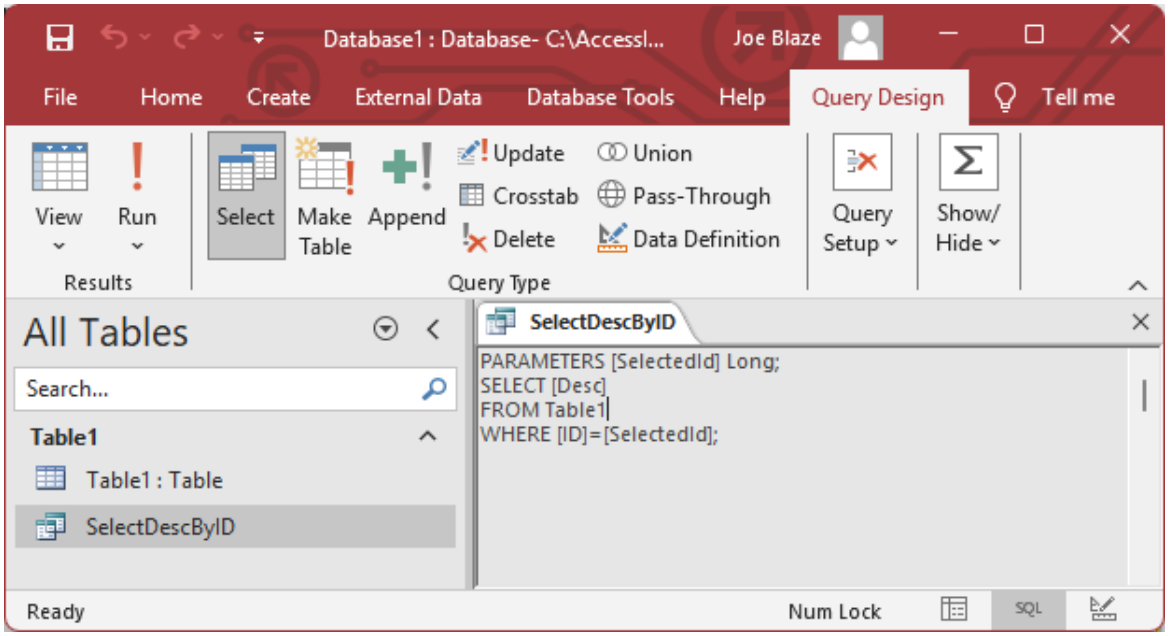
For Microsoft Access using the Ace ODBC driver:

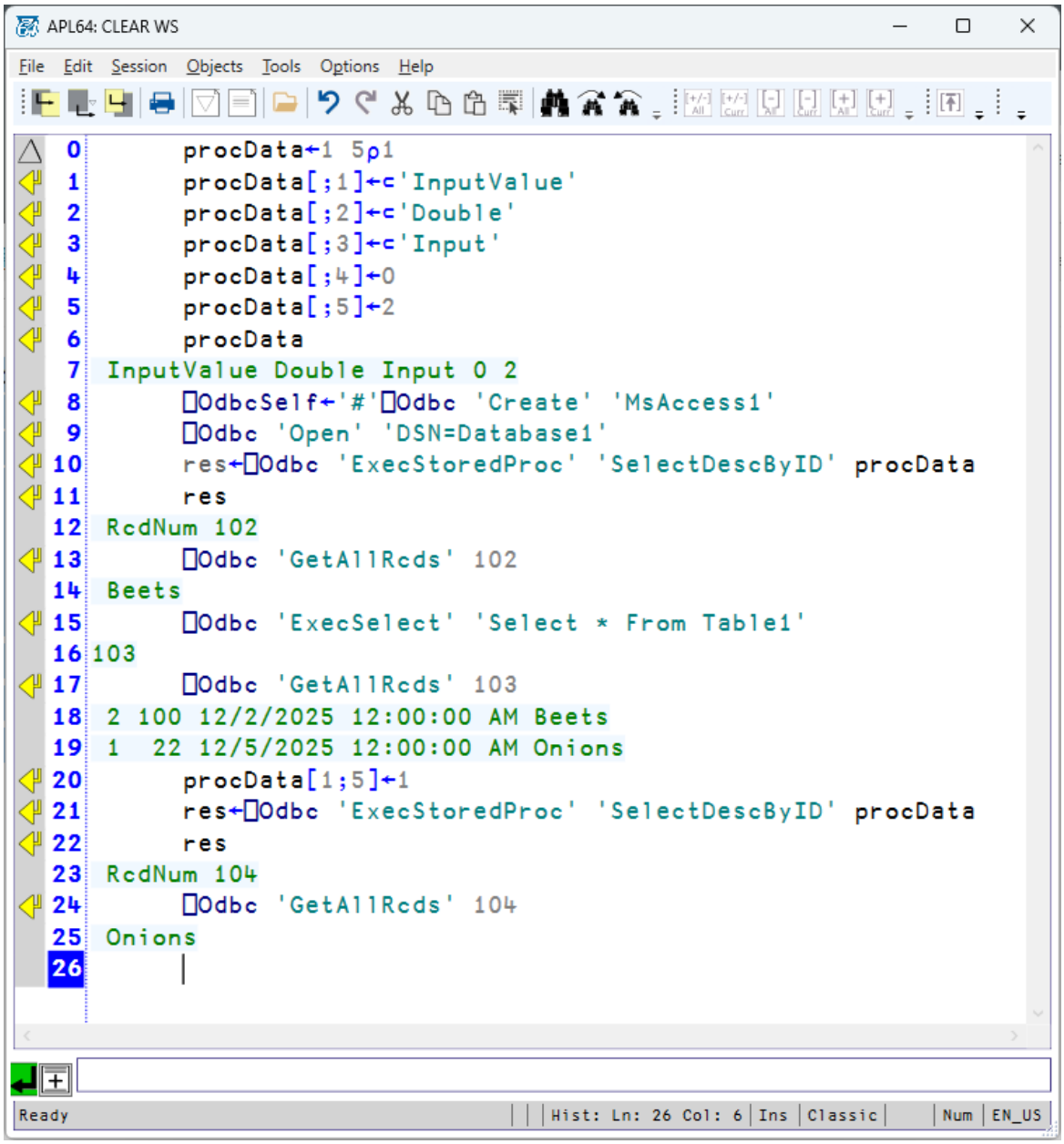
- The first column is the character vector 'RcdNum'
- The second column is the record set number which may be used to access the result data using the ⎕ODBC GetAllRecords or GetRecord actions.

For other database software:

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

Example: Microsoft Access Stored Query with an input parameter:

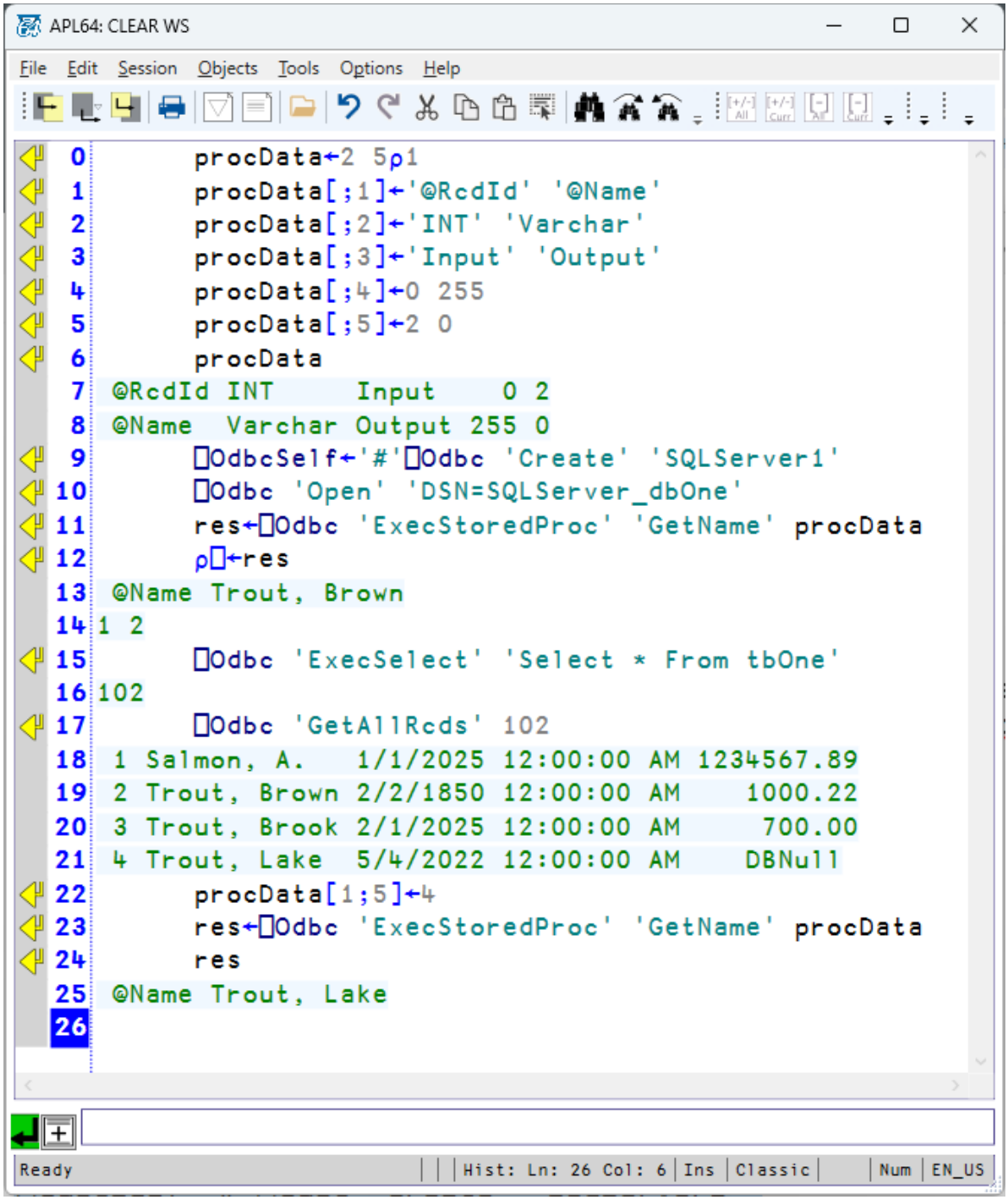




Example: Microsoft SQL Server stored procedure with input and output parameters:

Exec Action used to create the stored procedure:

```
CREATE PROCEDURE
[dbo].[GetName]
@RcdId INT,
@Name Varchar(255) OUTPUT
AS
SELECT @Name=Name
FROM tbOne
WHERE  RcdId=@RcdId;
```



GetAllRecords

Synonym: GetAllRcds

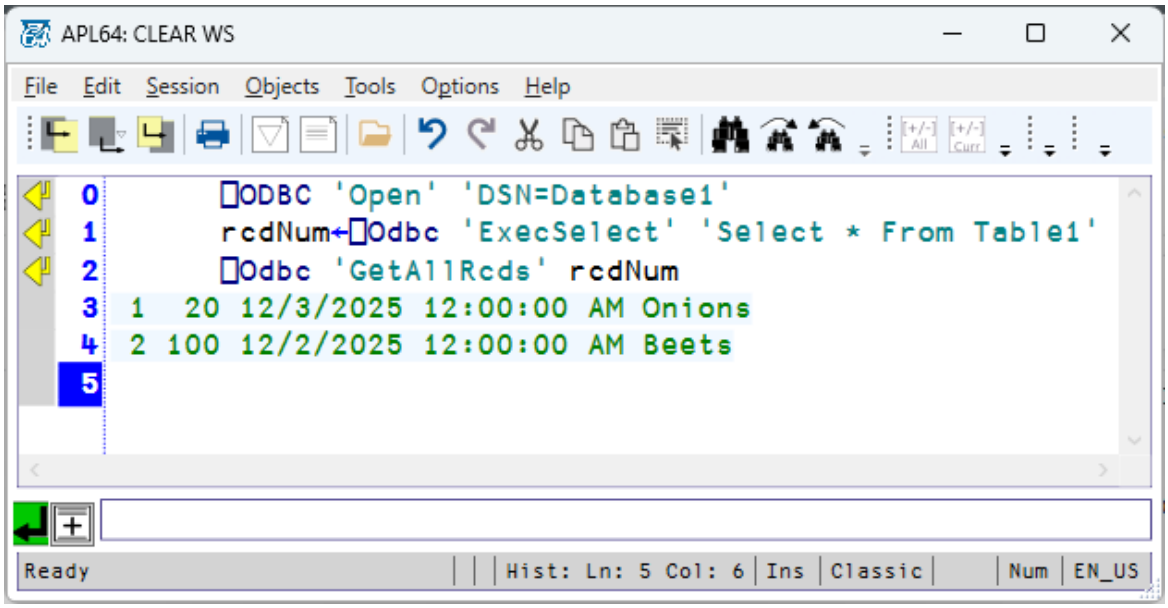
This action applies to an ⎕ODBC instance. The 'GetAllRecords' action will get the remaining records which have not already been read from the specified record set. This action closes the specified record set, because all records have been read.

Syntax: rcdsMatrix← [instanceName] ⎕Odbc 'GetAllRcds' rcdNum

An exception may occur if 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.

```
⎕ODBC 'Open' 'DSN=Database1'
rcdNum←⎕Odbc 'ExecSelect' 'Select * From Table1'
⎕Odbc 'GetAllRcds' rcdNum
```

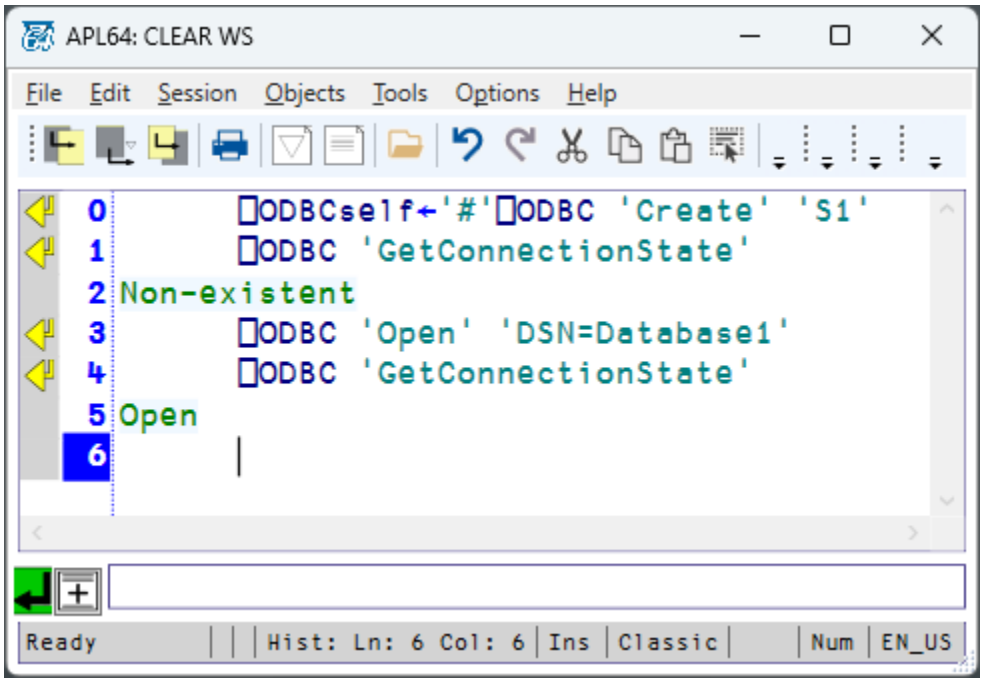


GetConnectionState

This action applies to an ⎕ODBC instance. The result of the GetConnectionState action is a text vector indicating the connection state of the specified ⎕ODBC instance to an Odbc database.

Syntax: result←[instanceName] ⎕ODBC 'GetConnectionState'

```
⎕ODBCself←'#'⎕ODBC 'Create' 'S1'
⎕ODBC 'GetConnectionState'
⎕ODBC 'Open' 'DSN=Database1'
⎕ODBC 'GetConnectionState'
```

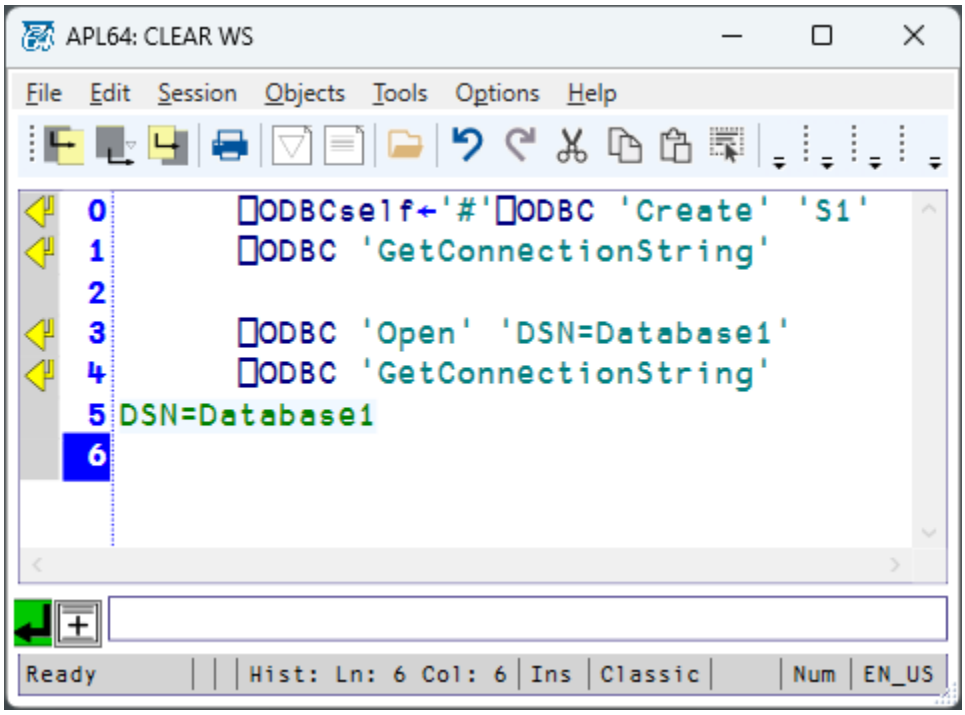


GetConnectionString

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

Syntax: result←[instanceName] ⎕ODBC 'GetConnectionString'

```
⎕ODBCself←'#'⎕ODBC 'Create' 'S1'
⎕ODBC 'GetConnectionString'
⎕ODBC 'Open' 'DSN=Database1'
⎕ODBC 'GetConnectionString'
```



GetRecord

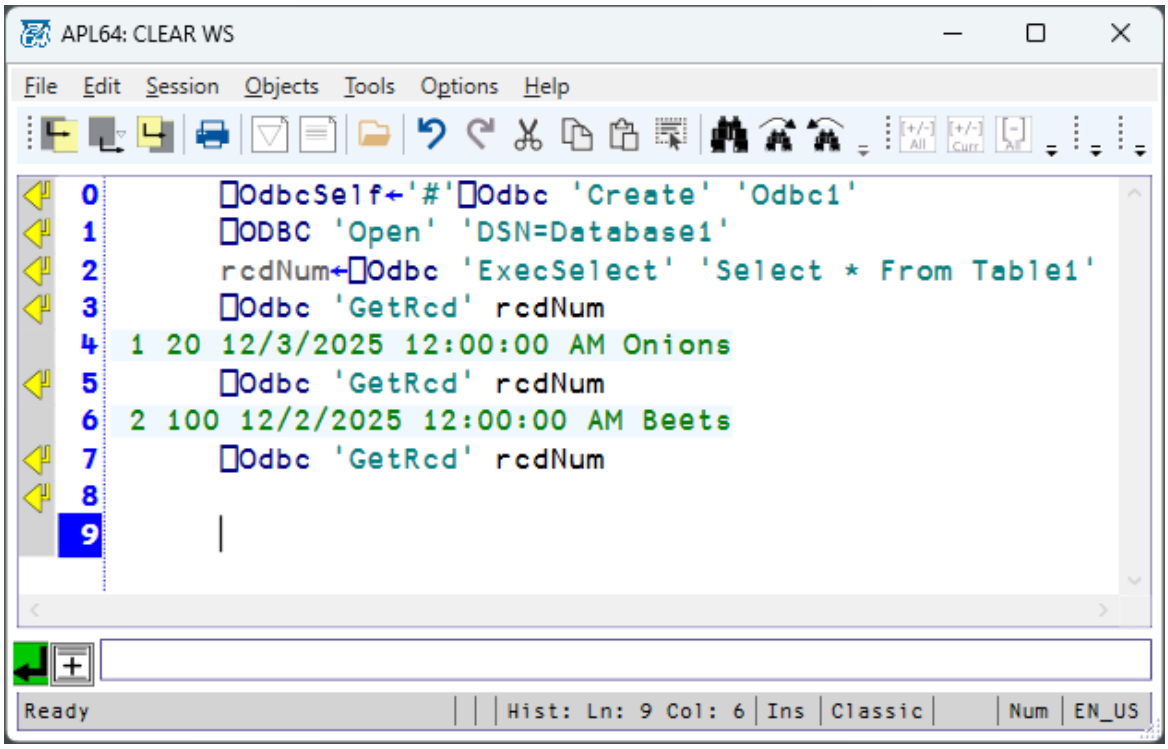
Synonym: GetRcd

This action applies to an ⎕ODBC instance. The right argument is the ⎕ODBC 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.

Syntax: rcd←[instanceName] ⎕Odbc 'GetRcd' rcdNum

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.

```
⎕OdbcSelf←'#'⎕Odbc 'Create' 'Odbc1'
⎕ODBC 'Open' 'DSN=Database1'
rcdNum←⎕Odbc 'ExecSelect' 'Select * From Table1'
⎕Odbc 'GetRcd' rcdNum
⎕Odbc 'GetRcd' rcdNum
⎕Odbc 'GetRcd' rcdNum
```



GetSchema

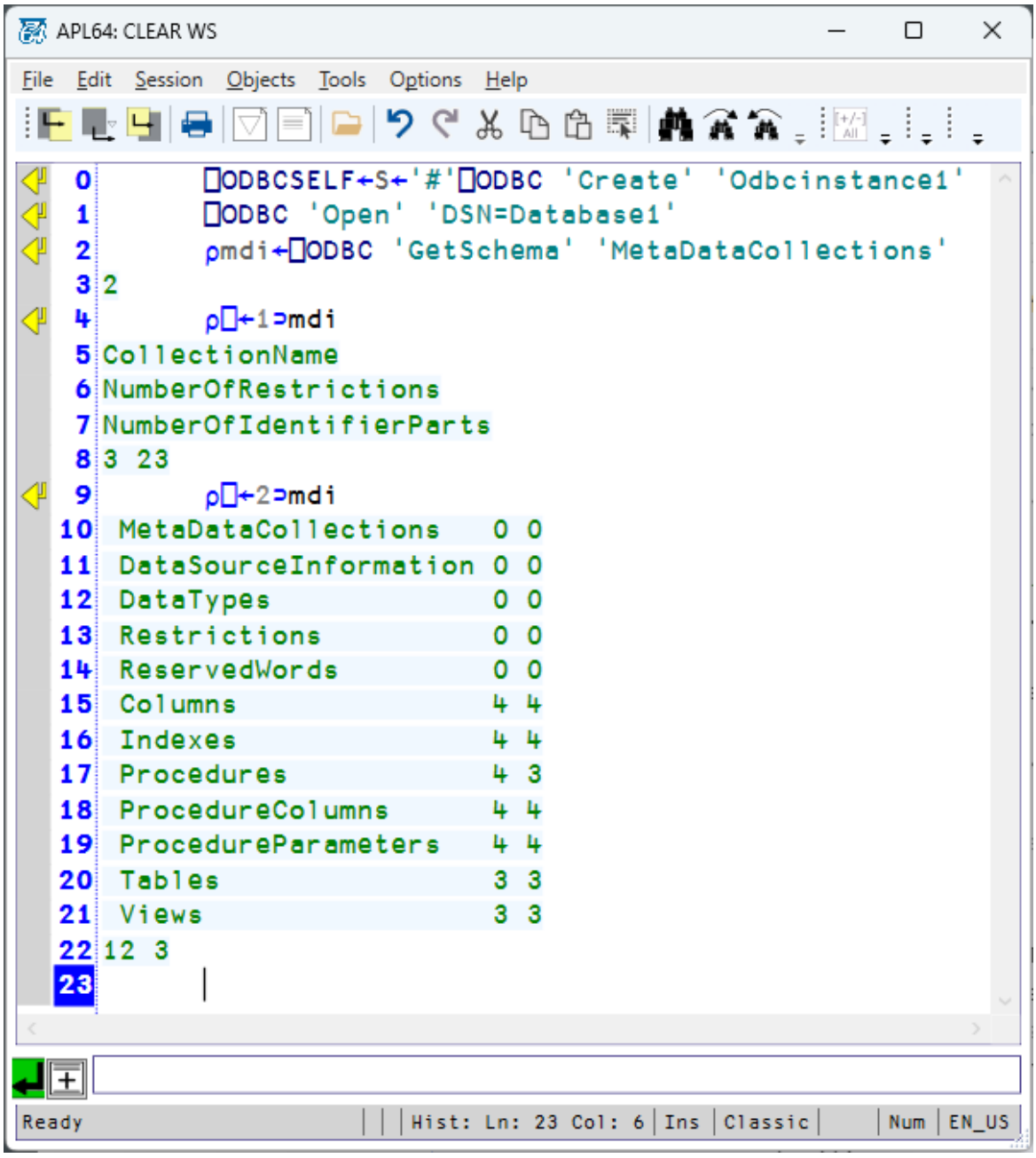
This action returns ODBC database metadata. The content and format of the OIDBC database metadata will vary with the specific database. Some metadata collection elements may have no representation in APL64.

Syntax: (colNames collInfo) ←⎕ODBC 'GetSchema' 'metaDataCollectionName'

metaDataCollectionName: Text name of the requested metadata collection. For all the available metadata collections: ⎕ODBC 'GetSchema' 'MetadataCollections'
colNames is a character matrix containing the column names of the metadata collInfo matrix .
collInfo is a matrix containing the metadata results.

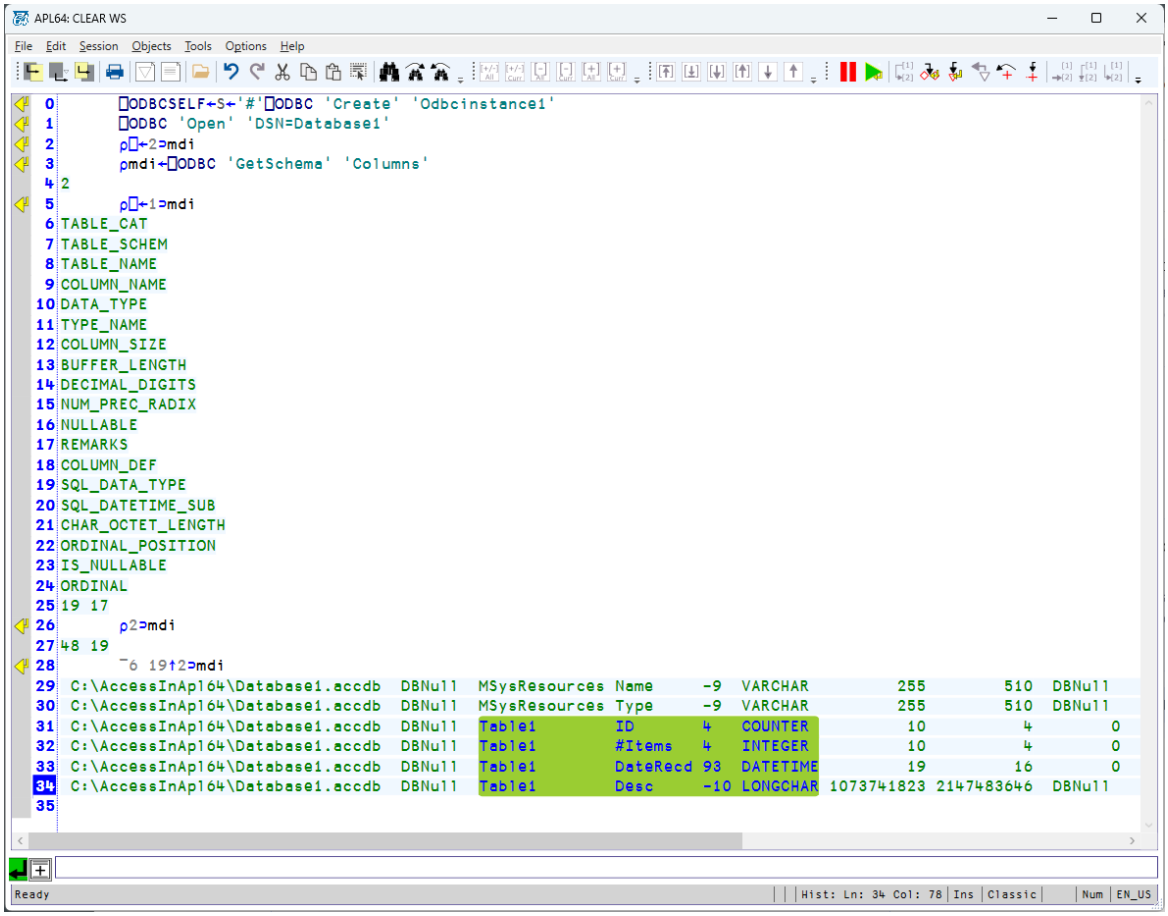
Example: MetaDataCollections collection in Microsoft Access

```
⎕ODBCSELF←S←'#'⎕ODBC 'Create' 'Odbcinstance1'
⎕ODBC 'Open' 'DSN=Database1'
⍴mdi←⎕ODBC 'GetSchema' 'MetaDataCollections'
⍴⎕←1⊃mdi
⍴⎕←2⊃mdi
```

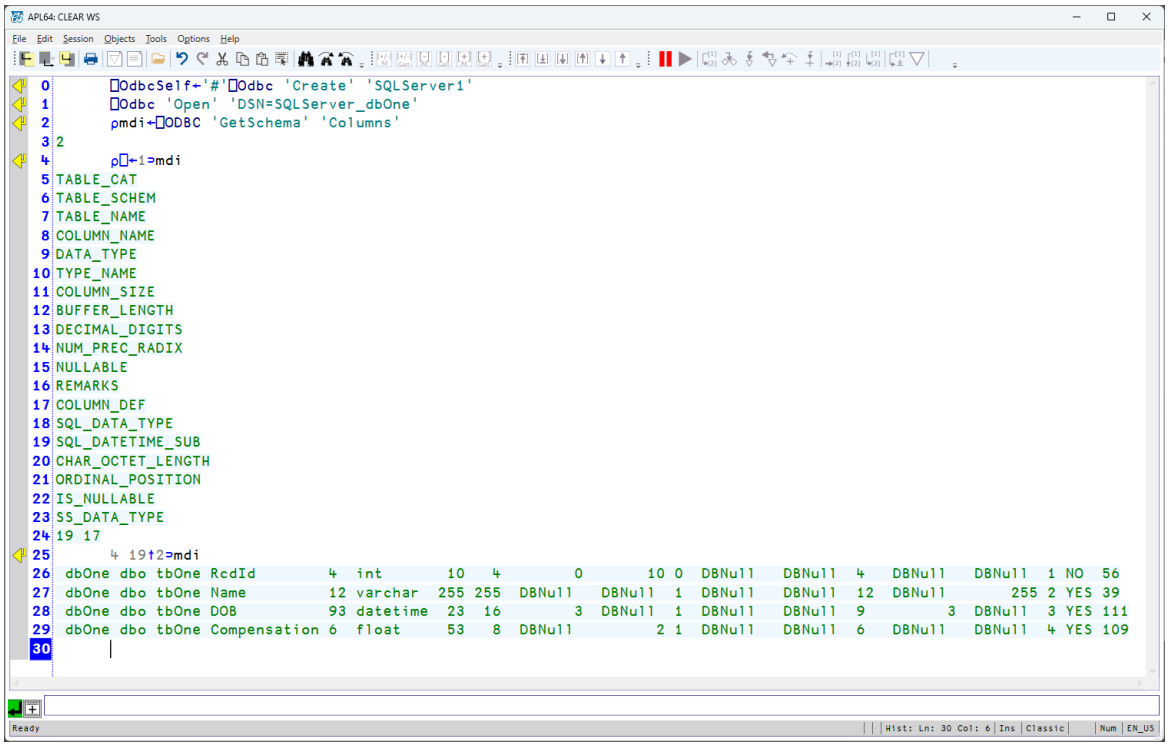


Example: Columns collection in Microsoft Access

```
⎕ODBCSELF←S←'#'⎕ODBC 'Create' 'Odbcinstance1'
⎕ODBC 'Open' 'DSN=Database1'
ρ⎕←2⊃mdi
ρmdi←⎕ODBC 'GetSchema' 'Columns'
ρ⎕←1⊃mdi
ρ2⊃mdi
¯6 19↑2⊃mdi
```

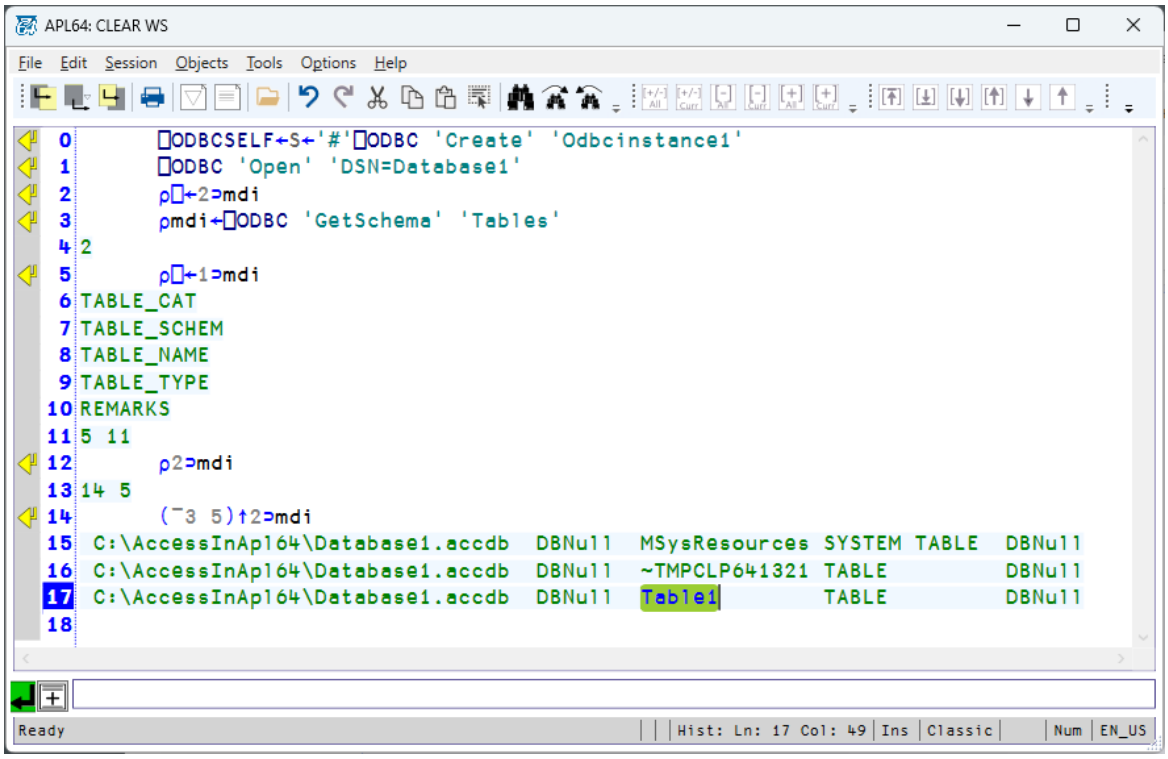


Example: Columns Collection in Microsoft SQL Server

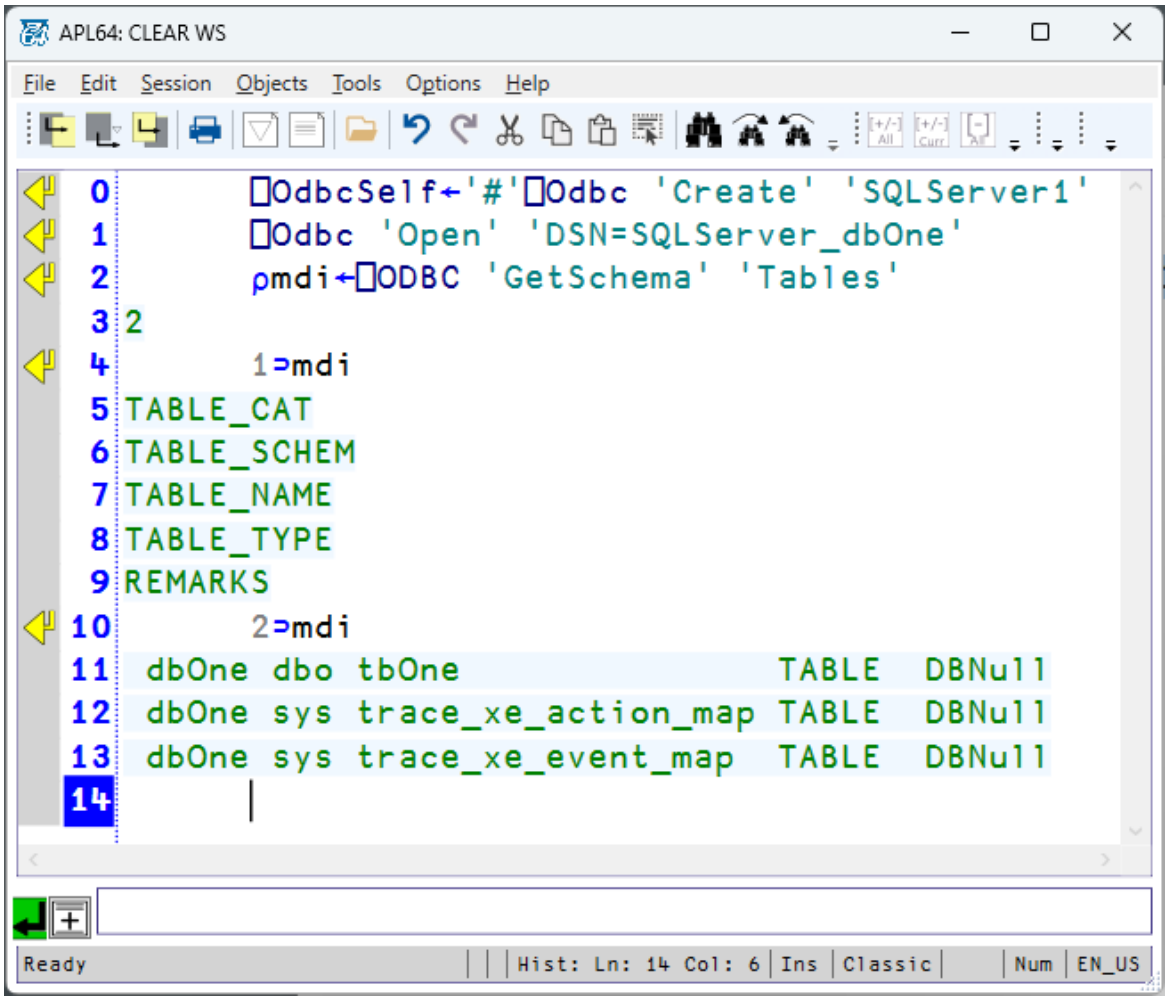


Example: Tables collection in Microsoft Access

```
⎕ODBCSELF←S←'#'⎕ODBC 'Create' 'Odbcinstance1'
⎕ODBC 'Open' 'DSN=Database1'
ρ⎕←2⊃mdi
ρmdi←⎕ODBC 'GetSchema' 'Tables'
ρ⎕←1⊃mdi
ρ2⊃mdi
(¯3 5)↑2⊃mdi
```



Example: Tables collection in Microsoft SQL Server

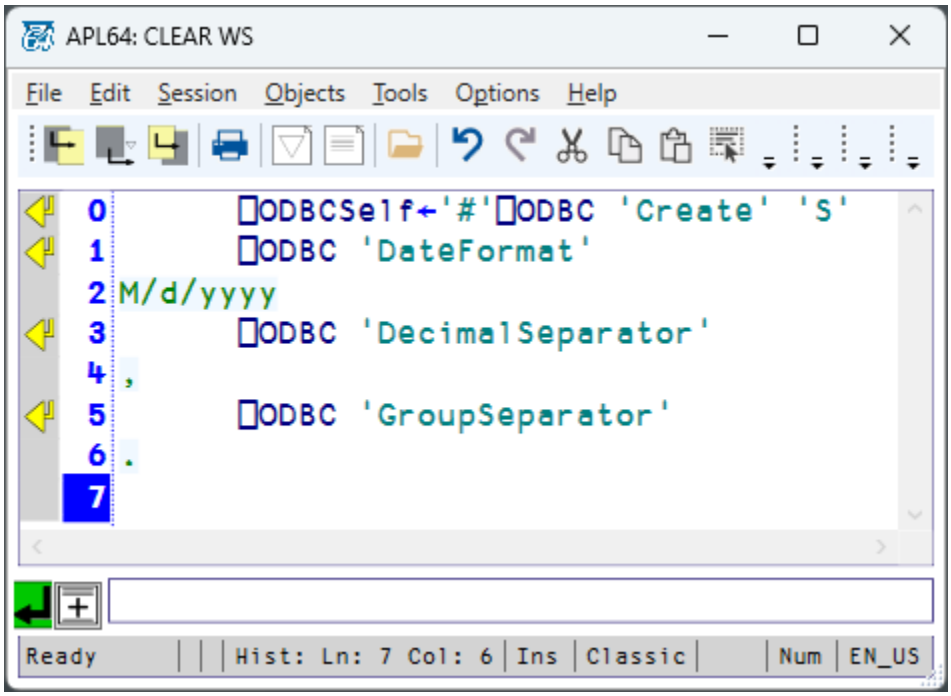


GroupSeparator

This action returns the .Net Number Group Separator text applicable to the specified ⎕ODBC instance.

Syntax: result←[instanceName] ⎕ODBC 'GroupSeparator'

```
⎕ODBCSelf←'#'⎕ODBC 'Create' 'S'
⎕ODBC 'DateFormat'
⎕ODBC 'DecimalSeparator'
⎕ODBC 'GroupSeparator'
```

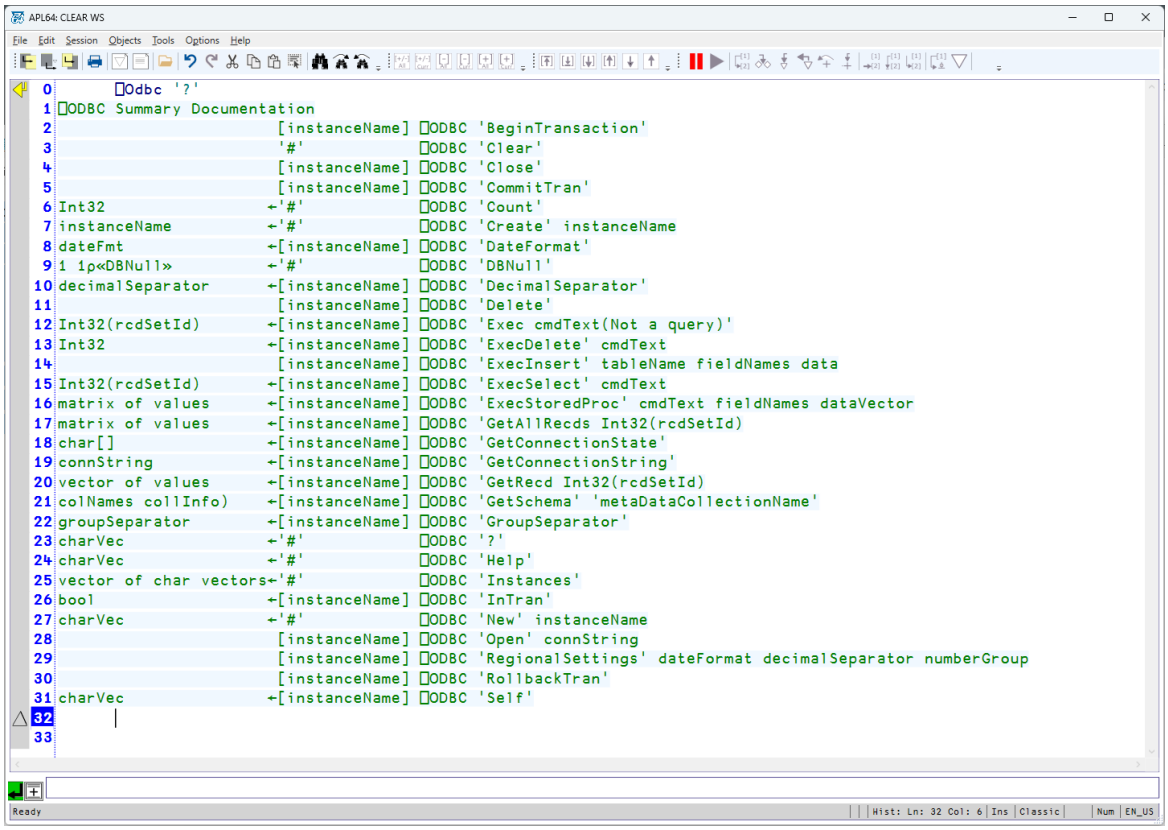


? or Help

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

Syntax: ⎕ODBC '?'

```
⎕ODBC '?'
```

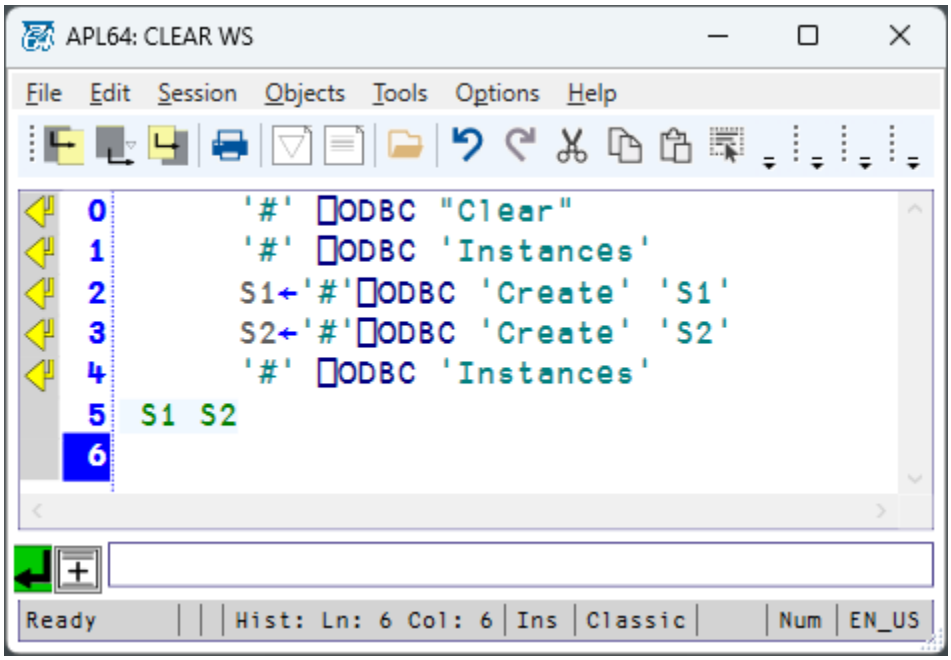


Instances

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

Syntax: result ← '#' ⎕ODBC 'Instances'

```
'#' ⎕ODBC "Clear"
'#' ⎕ODBC 'Instances'
S1←'#'⎕ODBC 'Create' 'S1'
S2←'#'⎕ODBC 'Create' 'S2'
'#' ⎕ODBC 'Instances'
```



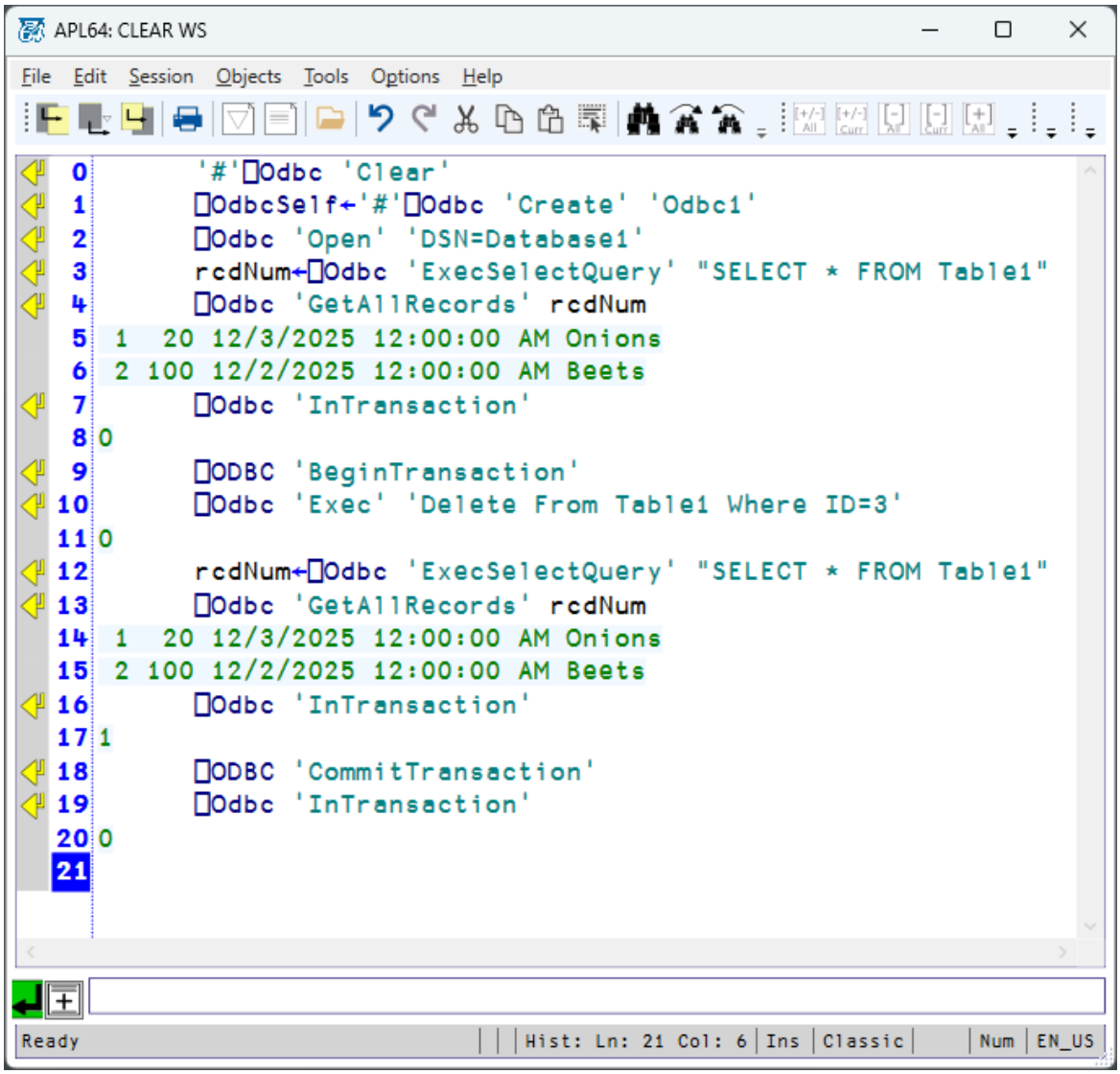
InTransaction

Synonym: InTran

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

Syntax: bool←[instanceName ⎕Odbc 'InTransaction'

```
'#'⎕Odbc 'Clear'
⎕OdbcSelf←'#'⎕Odbc 'Create' 'Odbc1'
⎕Odbc 'Open' 'DSN=Database1'
rcdNum←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdNum
⎕Odbc 'InTransaction'
⎕ODBC 'BeginTransaction'
⎕Odbc 'Exec' 'Delete From Table1 Where ID=3'
rcdNum←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdNum
⎕Odbc 'InTransaction'
⎕ODBC 'CommitTransaction'
⎕Odbc 'InTransaction'
```

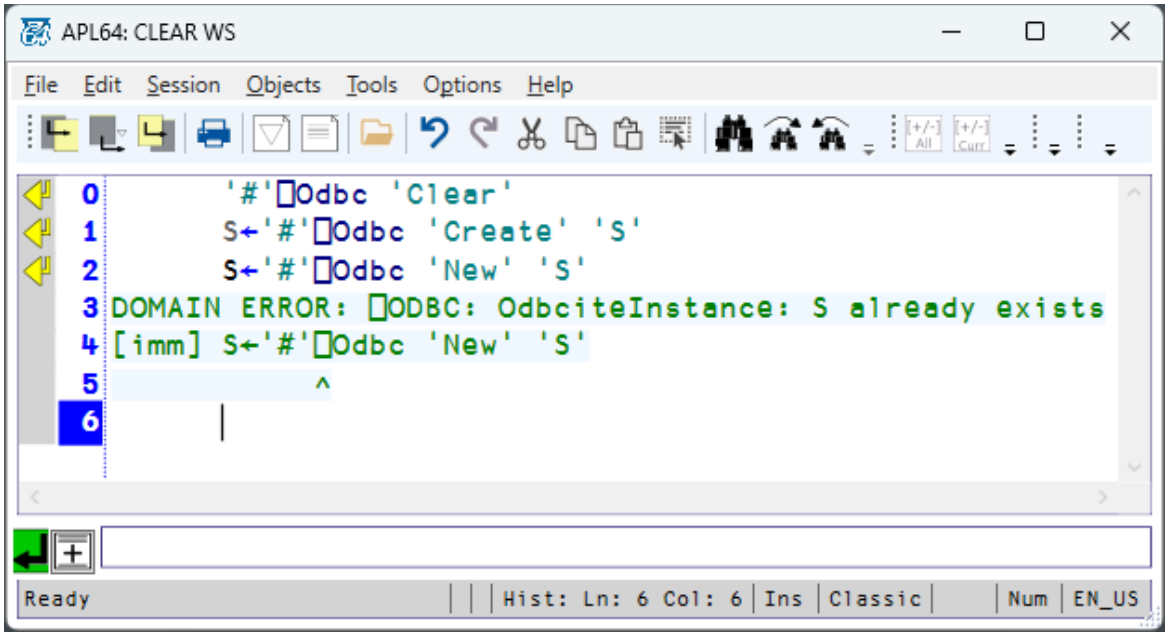


New

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

Syntax: instanceName←'#'⎕Odbc 'New' instanceName

```
'#'⎕Odbc 'Clear'
S←'#'⎕Odbc 'Create' 'S'
S←'#'⎕Odbc 'New' 'S'
```



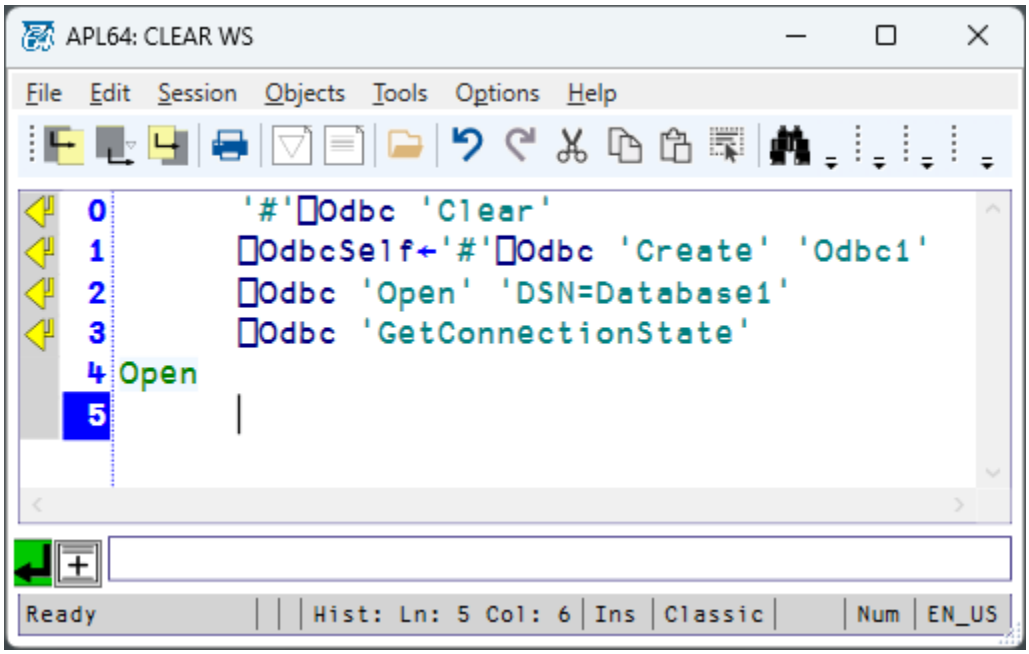
Open

This action applies to an ⎕ODBC instance. The Open action requires the specification of a connection string as the right argument.

Syntax: ⎕Odbc 'Open' connString

Important: After any access to a ⎕ODBC connection is concluded, use the ⎕ODBC 'Close' action to assure that access to the database is available to other database users.

```
'#'⎕Odbc 'Clear'
⎕OdbcSelf←'#'⎕Odbc 'Create' 'Odbc1'
⎕Odbc 'Open' 'DSN=Database1'
⎕Odbc 'GetConnectionState'
```

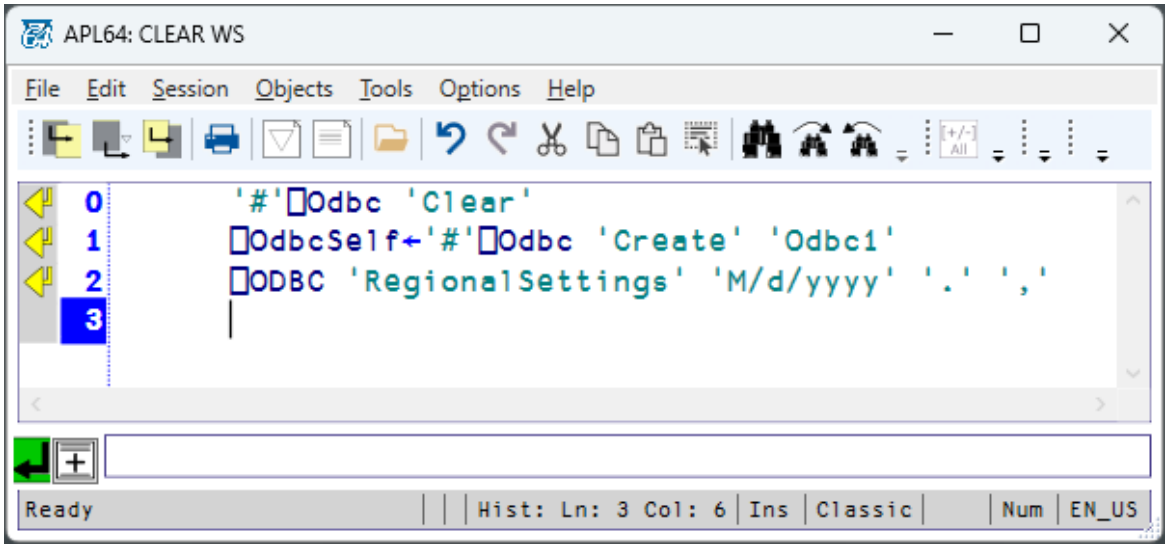


RegionalSettings

- This action can be used to set the regional values for the specified ⎕ODBC instance:
- Sdp: ShortDatePattern
- Cds: CurrencyDecimalSeparator, NumberDecimalSeparator & PercentDecimalSeparator
- Cgs: CurrencyGroupSeparator, NumberGroupSeparator & PercentGroupSeparator

Syntax: ⎕ODBC 'RegionalSettings' Sdp Cds Cgs

```
'#'⎕Odbc 'Clear'
⎕OdbcSelf←'#'⎕Odbc 'Create' 'Odbc1'
⎕ODBC 'RegionalSettings' 'M/d/yyyy' '.' ','
```



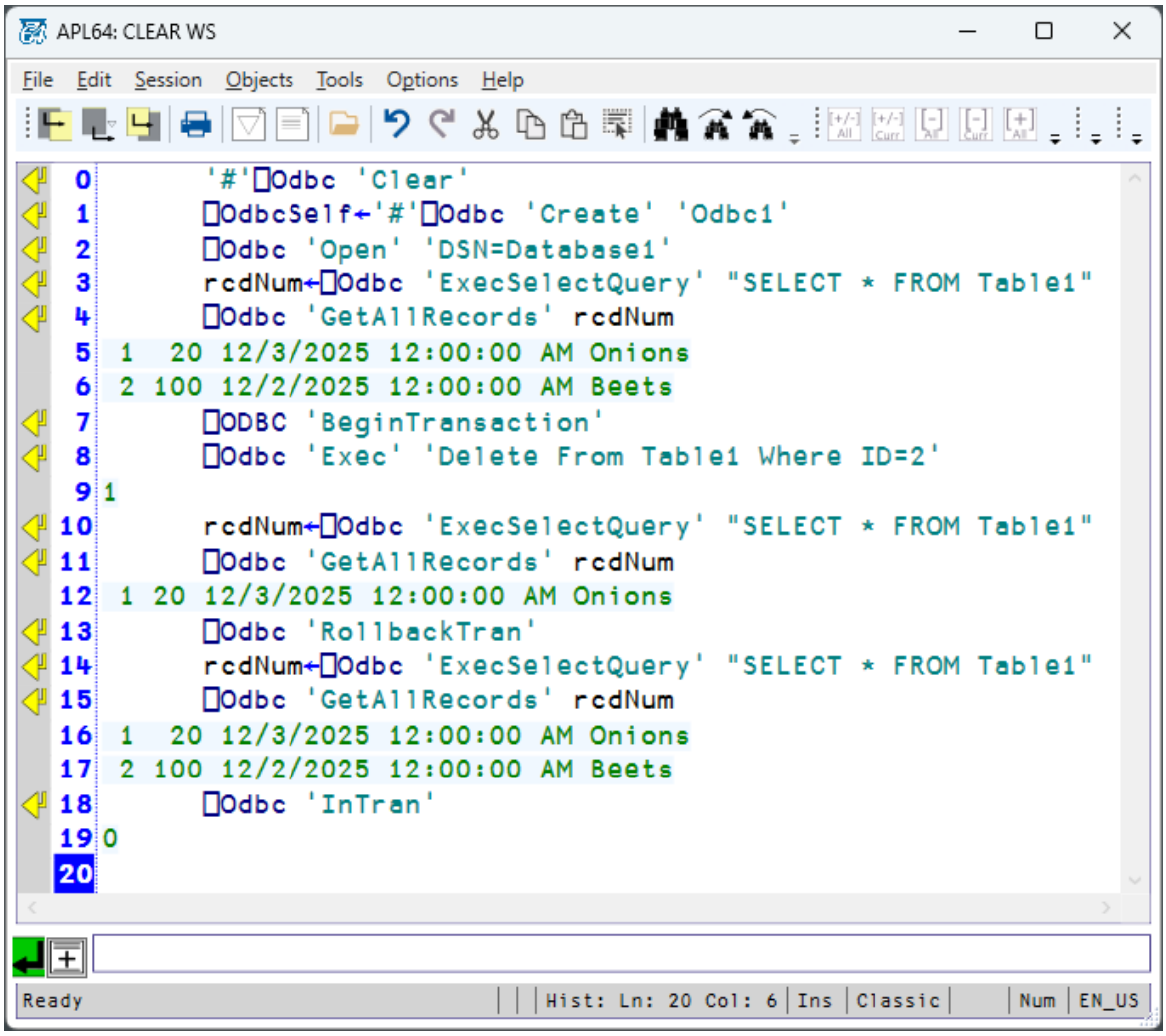
RollbackTransaction

Synonym: RollbackTran

This action applies to an ⎕ODBC instance. The RollbackTransaction action is used to cancel any pending ODBC operations which are included in the current ODBC transaction. The RollbackTransaction action has no result. The RollbackTransaction action has no effect if no ODBC transaction is in progress.

Syntax: ⎕Odbc 'RollbackTran'

```
'#'⎕Odbc 'Clear'
⎕OdbcSelf←'#'⎕Odbc 'Create' 'Odbc1'
⎕Odbc 'Open' 'DSN=Database1'
rcdNum←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdNum
⎕ODBC 'BeginTransaction'
⎕Odbc 'Exec' 'Delete From Table1 Where ID=2'
rcdNum←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdNum
⎕Odbc 'RollbackTran'
rcdNum←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdNum
⎕Odbc 'InTran'
```

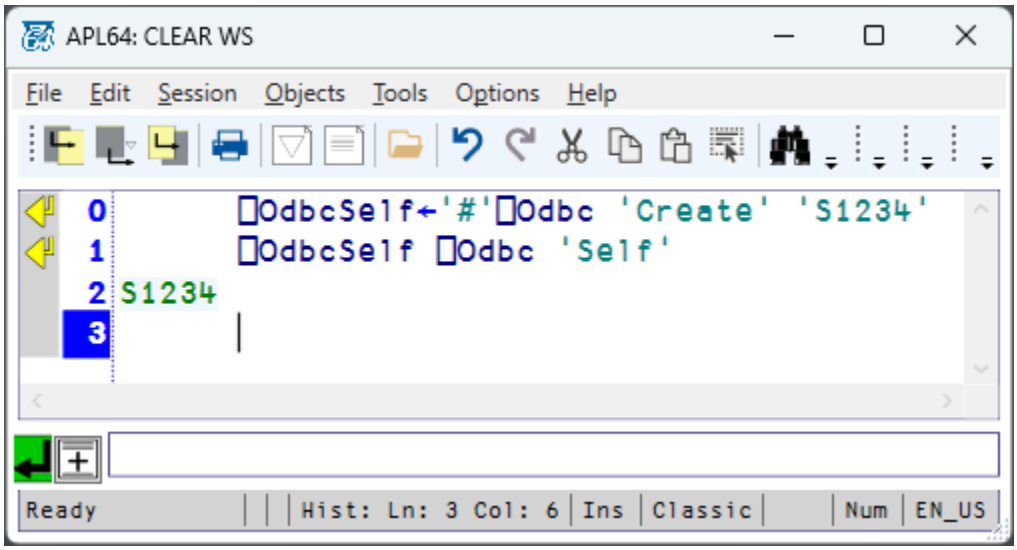


Self

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

Syntax: instanceName←'#'⎕Odbc 'Self' instanceName

```
⎕OdbcSelf←'#'⎕Odbc 'Create' 'S1234'
⎕OdbcSelf ⎕Odbc 'Self'
```



Learn Structured Query Language

http://www.w3schools.com/Odbc/default.asp

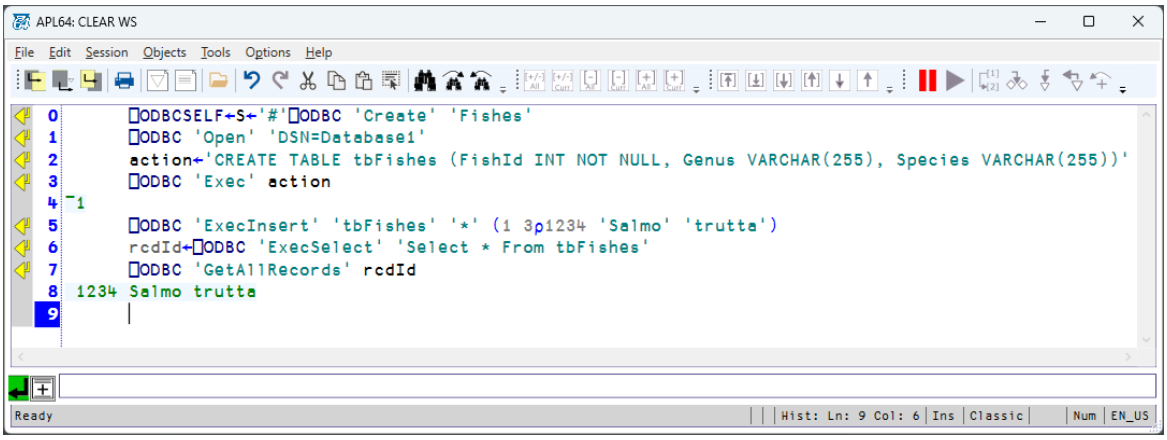
Some Additional Examples

Create Table

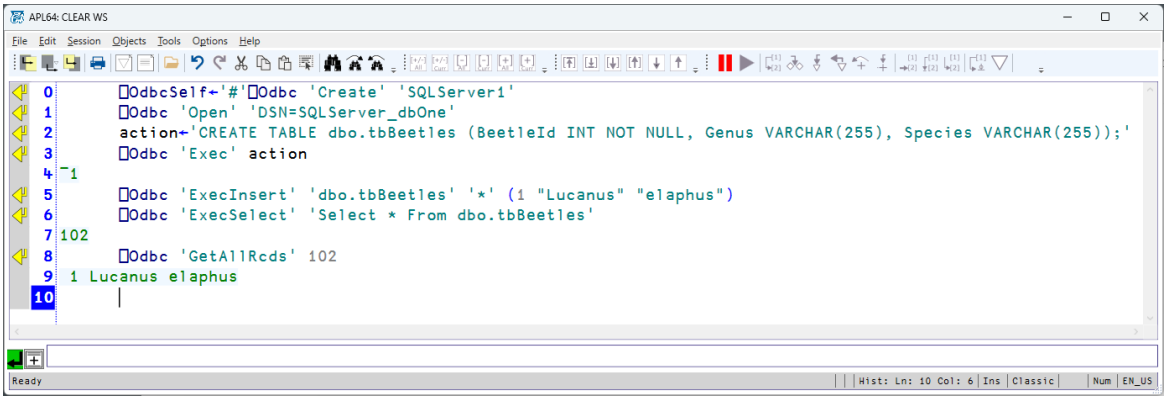
Use the ⎕ODBC 'Exec' action to create a new table within an existing database using an appropriate ODBC statement. The result (¯1) does not indicate an exception has occurred.

Example: Create Table in Microsoft Access

```
⎕ODBCSELF←S←'#'⎕ODBC 'Create' 'Fishes'
⎕ODBC 'Open' 'DSN=Database1'
action←'CREATE TABLE tbFishes (FishId INT NOT NULL, Genus VARCHAR(255), Species VARCHAR(255))'
⎕ODBC 'Exec' action
⎕ODBC 'ExecInsert' 'tbFishes' '*' (1 3⍴1234 'Salmo' 'trutta')
rcdId←⎕ODBC 'ExecSelect' 'Select * From tbFishes'
⎕ODBC 'GetAllRecords' rcdId
```



In Microsoft SQL Server

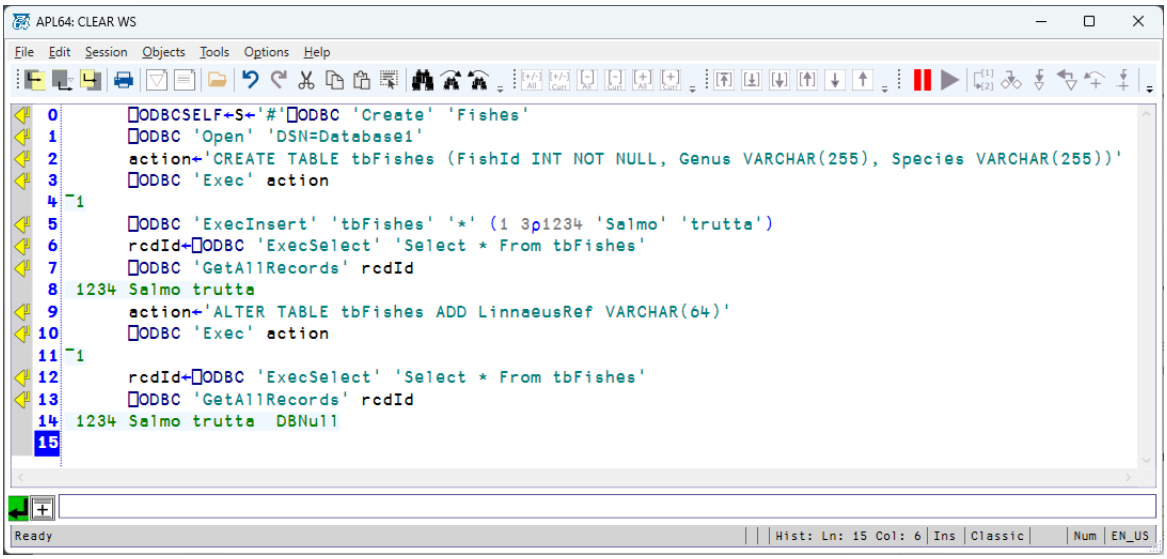


Alter Table

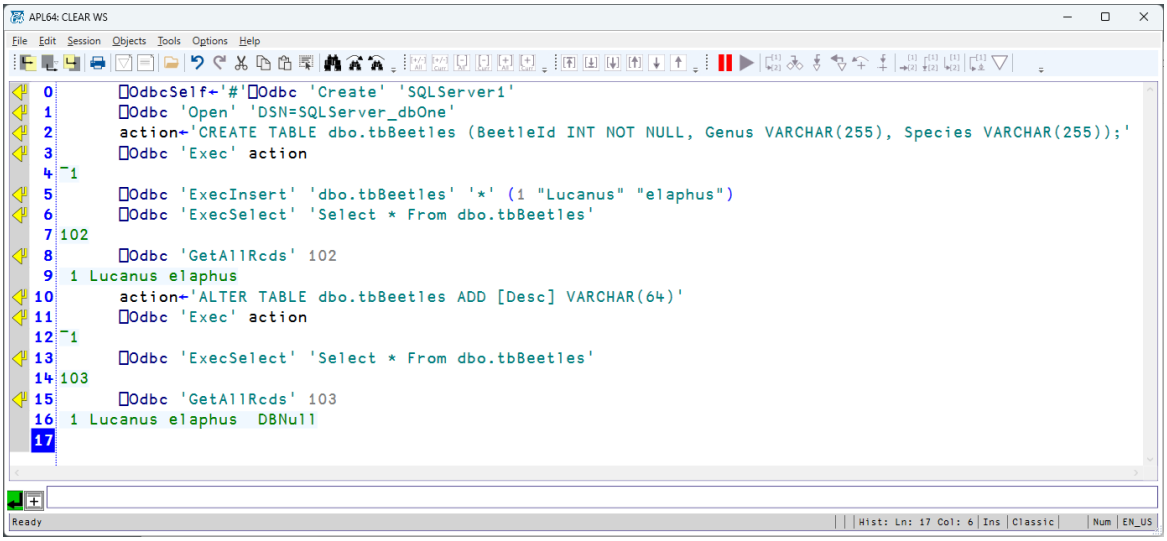
Use the ⎕ODBC 'Exec' action with an Alter Table ODBC statement to modify an existing table in an existing database. In this example, a new column is added to an existing table using an appropriate ODBC statement. The result (¯1) does not indicate an exception has occurred.

Example: Alter Table in Microsoft Access

```
⎕ODBCSELF←S←'#'⎕ODBC 'Create' 'Fishes'
⎕ODBC 'Open' 'DSN=Database1'
action←'CREATE TABLE tbFishes (FishId INT NOT NULL, Genus VARCHAR(255), Species VARCHAR(255))'
⎕ODBC 'Exec' action
⎕ODBC 'ExecInsert' 'tbFishes' '*' (1 3⍴1234 'Salmo' 'trutta')
rcdId←⎕ODBC 'ExecSelect' 'Select * From tbFishes'
⎕ODBC 'GetAllRecords' rcdId
action←'ALTER TABLE tbFishes ADD LinnaeusRef VARCHAR(64)'
⎕ODBC 'Exec' action
rcdId←⎕ODBC 'ExecSelect' 'Select * From tbFishes'
⎕ODBC 'GetAllRecords' rcdId
```



In Microsoft SQL Server,

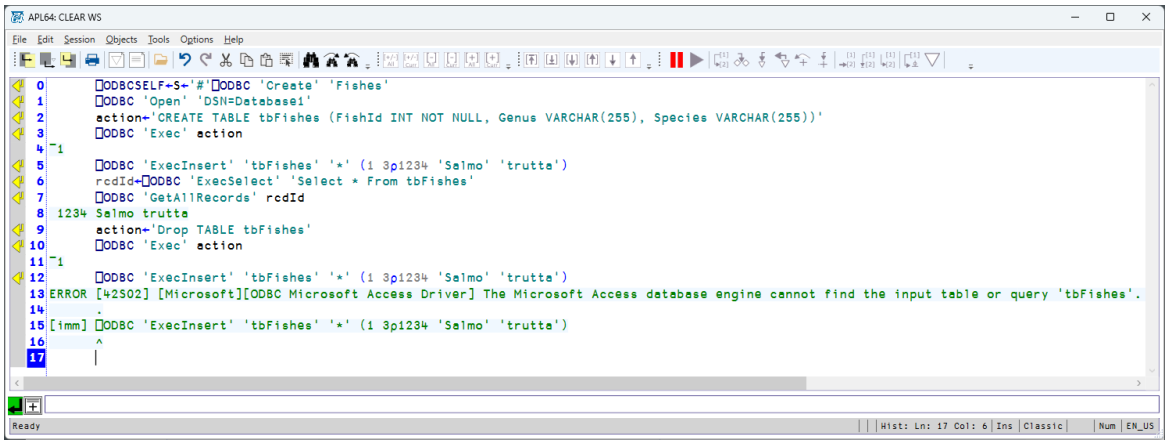


Drop Table

Use the ⎕ODBC 'Exec' action with a Drop Table statement. The result (¯1) does not indicate an exception has occurred.

```
⎕ODBCSELF←S←'#'⎕ODBC 'Create' 'Fishes'
⎕ODBC 'Open' 'DSN=Database1'
action←'CREATE TABLE tbFishes (FishId INT NOT NULL, Genus VARCHAR(255), Species VARCHAR(255))'
⎕ODBC 'Exec' action
⎕ODBC 'ExecInsert' 'tbFishes' '*' (1 3⍴1234 'Salmo' 'trutta')
rcdId←⎕ODBC 'ExecSelect' 'Select * From tbFishes'
⎕ODBC 'GetAllRecords' rcdId
action←'Drop TABLE tbFishes'
```

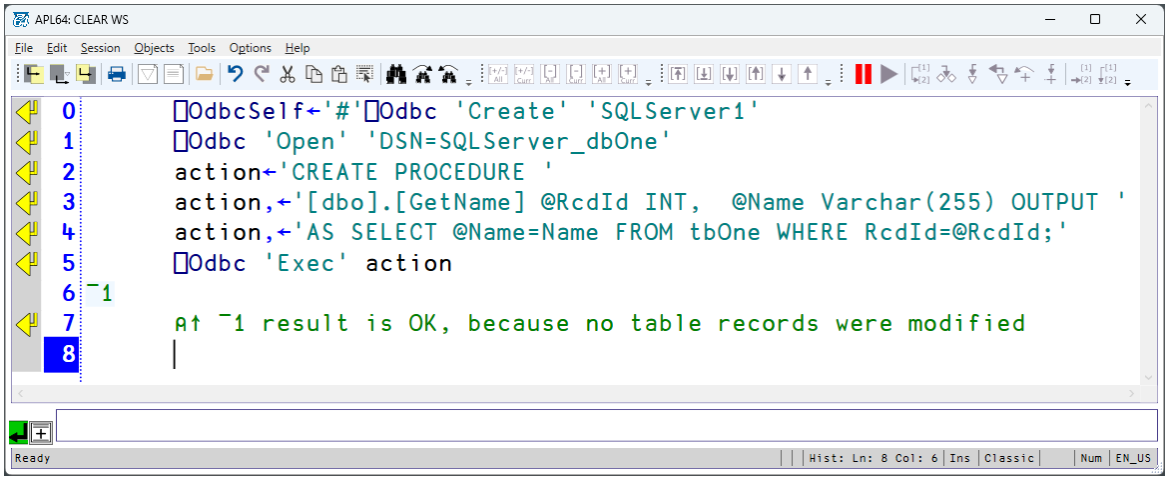
```
⎕ODBC 'Exec' action
⎕ODBC 'ExecInsert' 'tbFishes' '*' (1 3ρ1234 'Salmo' 'trutta')
```



Create/Alter Procedure

In Microsoft Access, it is not possible to create or alter a procedure at runtime using ODBC 'Create Procedure' or 'Alter Procedure'. If the Access database is linked to a Microsoft SQL Server database, the (stored) procedure can be created there, but this approach is beyond the scope of this document.

Example: Create/Alter Procedure in Microsoft SQL Server



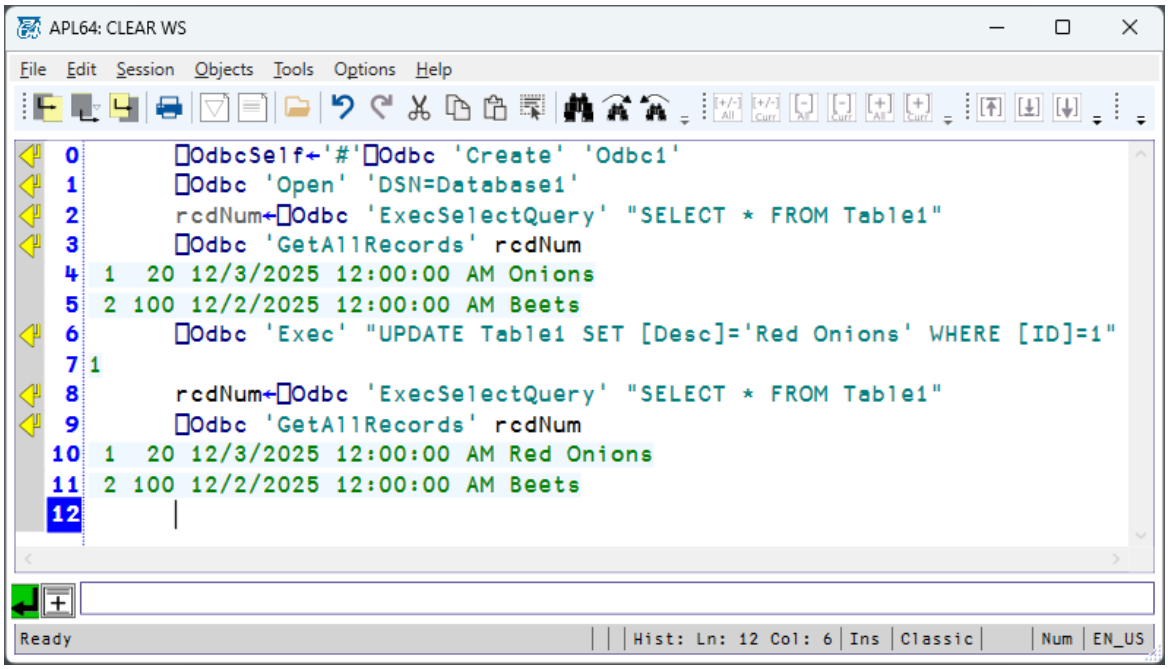
Update Table Record

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

Method #1:

Use the ⎕ODBC ‘Exec’ action with an appropriate ODBC UPDATE statement:

```
⎕OdbcSelf←'#'⎕Odbc 'Create' 'Odbc1'
⎕Odbc 'Open' 'DSN=Database1'
rcdNum←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdNum
⎕Odbc 'Exec' "UPDATE Table1 SET [Desc]='Red Onions' WHERE [ID]=1"
rcdNum←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdNum
```



Method #2:
Within an ODBC transaction, delete the desired table record and insert the modified table record

```
⎕OdbcSelf←'#'⎕Odbc 'Create' 'Odbc1'
⎕Odbc 'Open' 'DSN=Database1'
rcdNum←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdNum
⎕Odbc 'BeginTran'
⎕Odbc 'ExecDelete' 'Delete From Table1 Where [ID]=1'
rcdNum←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdNum
⎕Odbc 'InTran'
⎕Odbc 'ExecInsert' 'Table1' '*' (1 4⍴1 22 '#12/5/2025#' 'Onions')
⎕Odbc 'CommitTran'
rcdNum←⎕Odbc 'ExecSelectQuery' "SELECT * FROM Table1"
⎕Odbc 'GetAllRecords' rcdNum
```