Table of Contents

What's New in Version 5?

This document describes what is new in version **5.1.0.94** of the CLI based drivers compared to version 4.1.0.0.

Ignore_Case_Support and Uppercase Columns

Warning: This change may have consequences for existing databases / applications!!

The Ignore_Ucase_Support flag controls whether special uppercased columns (U_ columns) are created.

If Ignore_Ucase_Support is set to true (the new default), the Connectivity Kit will **not** create uppercased (U_) columns. Also, for existing tables having uppercase columns, the uppercased columns will be removed on restructure of the table.

The main reasons for changing the default to no longer create uppercase columns:

- The uppercase columns are only useful in some very specific circumstances. Especially: When a Case Insensitive collating sequence is used on the database backend, the uppercase columns are not needed.
- Use of uppercase columns has bad influence on performance.
- Uppercase columns can be confusing to users of external tools (e.g Crystal Reports).

Uppercase Columns

Uppercased (U_) columns were introduced to be fully compatible with the DataFlex embedded database.

In the DataFlex embedded database, we know the concept of uppercased index segments. This is different from other (SQL based) databases where this concept does not exist.

In the DataFlex embedded database columns appearing in an index can be marked uppercased (or case insensitive) or not. The same column can be uppercased when it is used as a segment in one index and non-uppercased when it is used as a segment in another index.

In other databases case sensitivity is defined on the column. (Not on the index segment). It is defined by the used collating sequence that usually can be Case Sensitive or Case Insensitive.

Although the handling of case (in)sensitivity is not exactly the same in the DataFlex embedded database and SQL based databases, in most situations this does not require the existence of special uppercase (U_) columns.

Specifically, when the database backend uses a ***case insensitive*** collating sequence (the default on most backends), the uppercase columns are not needed. During find operations the same records will be found. It makes no difference if these finds are based on the original columns or on the uppercased (U_) columns.

When using a ***case sensitive*** collating sequence on the backend, not having U_ columns may cause differences in behavior for existing applications. This will only be the case if the Ignore_Case flag for an index segment is on. (Df_Index_Segment_Case = Df_Case_Ignore).

Find operation may find the record with U_ columns, but not find it without U_ Columns:

Example: If a table has a row with 'AA':

Find eq 'aa'

will find the row if table has U_ columns, but will nor find it if the table has no U_ columns.

Note this behavior difference may also influence relates and constrains.

Consequences for Existing Databases

The change to no longer generate uppercased (U_) columns may have consequences for existing applications or databases.

If you have an existing database that has tables with uppercased (U_) columns, and you are upgrading from an earlier Connectivity Kit version: be aware of following consequences:

- The Connectivity Kit will no longer create uppercase (U_) columns during conversion or restructure.
- If existing tables with U_ columns are restructured, the U_ columns will be removed.
- Existing tables still having U_ columns will behave as before in Open, Find and Save operations. The U_ columns will be used as before in the SQL statements generated by these operations.
- The Ignore_case setting (Df_Index_Segment_Case attribute) will always be off. The flag has no meaning anymore. Case sensitivity is determined by the used collating sequence of the backend.
- If tables are used by external tools, removal of U_ columns may require changes. For example: In Crystal Reports a Verify Database is necessary after removing columns.
- When already using Case Insensitive collating sequence on the database backend, removal of U_ Columns will not cause existing programs to behave different.
- When using Case Sensitive collating sequence, existing applications may behave different after removal of U_ columns.

How to Keep Using Uppercase Columns

If you need to keep using uppercase columns, the Ignore_Ucase_Support can be set to false in the driver configuration file. Or the Df_Driver_Ignore_UCase_Support or Df_Database_Ignore_UCase_Support attributes can be set to false at runtime.

If Ignore_Ucase_Support is set to false, the Connectivity Kit will behave the same as earlier versions.

Speed changes

We have made several speed changes that improve find speed. We changed the cursor type that is used, switched to using block cursors, changed how we treat statement handles and implemented JIT binding.

Cursor type

Testing showed that the forward only cursor is faster then any other cursor type. The 5.x version of the CLI based Connectivity Kits will use the forward only cursor for every find done outside of a transaction. It will always use forward only cursors for tables that are set never to lock (DF_FILE_MODE set in such a way that the DF_FILE_MODE_NO_LOCKS bit is set). For any find operation done inside a transaction, we determine the cursor type to use in the same way as we used to do in version 4.1.

Block cursors

The 5.x version of the CLI based Connectivity Kits will get multiple records when it fetches data from a cursor. The number of records that will be fetched can be configured per table by using the intermediate file keyword ***Block_Size*** (minimal value = 2). Getting multiple records is not done for cursors created to handle a find equal operation. This will reduce the number of times the CK needs to communicate with the database server (server roundtrips).

Introducing block cursors speeds up find operations. It also creates a form of caching. The driver will get multiple records at a time and on every next find will get the record from memory rather then from the back end. A find cache timeout has been introduced. This timeout can be configured (default value 10 milliseconds), in milliseconds. When the time between two find operations is bigger then the timeout, the CK will fetch the data from the database even if not all records in the cache have been processed. The timeout can be configured using the ***Find_Cache_Timeout*** driver configuration file keyword.

The block size of a table can also be set by using the DF_FILE_BLOCK_SIZE table level attribute. This attribute can be set both inside and outside of a Structure_Start .. Structure_End operation.

Two table level attributes can be used to measure the effectiveness of the current find cache timeout setting. The attributes DF_FILE_FINDCACHE_HITS and

DF_FILE_FINDCACHE_TIMEOUTS will return the number of cache hits (usage of cache) and timeouts (discarding the cache and getting data from the server) respectively. The find logic will increment the appropriate attribute when needed. The attributes are kept in signed integer variables. Whenever incrementing the integer value of one of the attributes causes the value to become negative both variable will be set to 0 (zero). It is possible to set the attributes to 0 (zero, the only allowed value). Setting one of the attributes will automatically cause the other attribute to be set to 0 (zero).

Statement handles

The 5.x version of the CLI based Connectivity Kits will allocate two statement handles per table. One statement handle is used for all find equal operation on any index; the other one is used for all other find operations. The 4.1 CK would also allocate two handles but it would use one for find by recnum/rowid and the other for all other find operations (including all other find equal operations).

JIT Binding

To minimize network traffic we no longer get all columns of a table. If a column is big (default > 10 megabytes) its value will not be placed in the buffer in a normal find operation, the value will be placed in the buffer the first time the column value is asked for by the program.

JIT binding can be configured on two levels. On driver level one can set the threshold that makes a column to be eligible for JIT binding (default 10 Mb) using the ***JIT_Treshold*** keyword. On table level you can switch the use of JIT binding on or off (default on) using the ***JIT_Binding*** intermediate file keyword.

If JIT binding is switched off for a table, every column's value will be placed in the buffer on find operations..

JIT Binding can also be switched on and off by sing the DF_FILE_JIT_BINDING table level attribute. This attribute can only be set inside a Structure_Start .. Structure_End operation.

Lock logic changes

Since CLI is a record locking environment, changing find logic also influences how locking works.

Save (update) and delete

The save logic distinguishes between creating new records (insert) and changing existing records (update). The update and delete logic until version 5 was complicated. Depending on the back end capabilities and find operations done previously one of three methods was used:

- Positioned update/delete
- SQLSetPos update/delete
- Where rowid = current rowid

The first two methods depend on the current position of the cursor. Sine we now use block cursors, the current position of the cursor will most likely not be on the record being updated/deleted. Since we can no longer use these methods that depend on cursor position, we simplified the update/delete logic considerably. The 5.x version of the CLI based Connectivity Kits only uses the "where rowid = current rowid" method.

Locking records

In SQL environments the lock granularity usually is one record. The CLI drivers are designed to assume record locking it supported by the back end. Locking is part of the find logic (you can only lock a record if you find it). Not all back ends support the same lock functionality the way locking works differs per back end. If it is not possible to lock a record when finding, it will eventually get locked when it is saved or deleted.

The lock logic is only used in a transaction. All the descriptions below assume a transaction has been initiated. The type of lock functionality depends on the type of cursor supported by the back end.

The **SQL Server** Connectivity Kit will lock records "as they are found" inside a transaction no matter what find operation is done.

The **DB2** Connectivity Kit will lock a record when a find equal (on any index) is done inside a transaction. No lock is issued on other find operations. If an unlocked record is updated, it will be locked at the moment of the update (save).

The **ODBC** Connectivity Kit will get information from the back end on what lock functionality is supported for the database cursor types that are supported. It will choose the most appropriate strategy for the environment. There are three possible ways locking during find operations will work when using ODBC:

1. No locks are set during find operations. If a record is updated, it will be locked at the moment of the update (save).
2. Only find equal operations set a lock. If an unlocked record is updated, it will be locked at the moment of the update (save).
3. All find operations set a lock

We tested a number of back ends on the supported lock strategy. The result can be found in the table below:

Back end	Lock strategy
MySQL	2
Oracle	3
PostgreSQL	2
Microsoft Access	3

Deadlocks/Lock timeouts

The 5.x version of the CLI based Connectivity Kits will translate a deadlock or lock timeout error to the DataFlex lock timeout error DFERR_LOCK_TIMEOUT (4106). This error triggers a build in retry mechanism when explicit transactions are used.

If the DataFlex runtime detects the DFERR_LOCK_TIMEOUT error it will retry the transaction by jumping to the ***Lock***/***Reread***/***Begin_Transaction*** command that started the transaction. The retry will be attempted a number of times, the number of times can be set by using the ***Set_Transaction_Retry*** command it can be queried by using the ***Get_Transaction_Retry*** command. A retry consist of a two step process, first the function ***Verify_Retry*** is called; if the return value is 0 (zero), the transaction is retried by jumping to the ***Lock***/***Reread***/***Begin_Transaction*** command; if the return value is non zero the transaction will be aborted and logic will jump to the ***Unlock***/***End_Transaction*** command. The function will be called in the object that "contains" the transaction i.e the object that issued the ***Lock***/***Reread***/***Begin_Transaction***.

Until version 5.x we did not generate the DFERR_LOCK_TIMEOUT error. Reasoning behind this was that in a record locking environment a deadlock or a lock timeout can occur at any moment. If the transaction relies on program variables that change during the transaction, retrying can be dangerous. Before the retry the table buffers and their status are reset, program variables however are not reset. Over time we came to realize that transaction that rely on program variables are very rare. This is why version 5.x by default generates the DFERR_LOCK_TIMEOUT error. If a program does rely on changing program variables in a transaction it can be adjusted to use the automatic retry mechanism. Create a Verify_Retry function (Function Verify_Retry Returns Integer) that resets the program variables.

The translation can be switched off by setting the ***Use_DF_LockError*** keyword in the driver configuration file. When switched off error CLIERR_DEADLOCK_OR_TIMEOUT (12303) will be generated.

The version 5.x ODBC Connectivity Kit allows the user to setup a translation list of deadlock and lock timeout errors or SQL States in the database level configuration file. Previous versions of the ODBC CK did not offer any form of translation of this type of error. An error number list can be created using the ***Native_LockError*** keyword; a SQL State list can be created using the ***Lock_State*** keyword.

Login changes

Silent login

The login logic used a function that caused the database client software to popup a login panel if the information supplied was incomplete or wrong. The pop up can be very convenient because it allows the information in the server identification string to be minimal, individual users can complete the information as they login.

The panel makes the login an interactive process. There are situations where this is not desired behavior.

In the driver configuration file the ***Silent_Login*** keyword can be set to switch off the popup panel. If Silent_Login is on and the login information in incomplete or wrong, a login error will be generated and no panel will pop up.

Redirect connection

A new function in the cCLIHandler class allows redirecting an existing connection. When used the connection will change (point to another database) but the tables stay open. This is intended to be used in cases where a program must switch between databases that are identical in definition.

The function gets two string arguments, sOldConnection and sNewConnection. It will check if the original connection exists, login to the new connection, redirect the open tables and logout from the old connection. The function returns 0 when the redirection was successful. A return value of 1 means the original connection does not exist. A return value of 2 means logging in to the new connection failed. The arguments should contain the string that identifies the database connection. This is usually what can be found in the DF_FILE_LOGIN attribute of a table.

If for example we want to redirect the connection to an SQL Server database from database DBX to database DBY both on database server S we would use:

```
Get RedirectConnection "SERVER=S;Trusted_Connection=yes;DATABASE=DBX" ;
              "SERVER=S;Trusted_Connection=yes;DATABASE=DBY" ;
              To iResult
```

A connection can only be redirected if all tables that are open in the original connection are also present with an identical definition in the new connection.

Automatic client selection (SQL Server)

A client server database environment uses two processes to handle database connections. A database server process runs the database and stores the actual data. A client process communicates with the server process to retrieve or modify data.

With the introduction of SQL Server 2005 Microsoft created a new Client. It is possible for a machine to have two different SQL Server clients installed; one for SQL Server 2000 and one for SQL Server 2005. A client can connect to database servers from a different version (even if that server has a newer version).

In the login logic of the Connectivity Kit, the client that is used to connect is chosen based on the installed client. If the SQL Server 2005 client is installed it will be used (even to connect to older versions of the database server), otherwise the SQL Server 2000 client is used.

DataFlex Connection IDs

We introduced a new concept to define a connection, the "DataFlex connection ID". Basically this provides a way to give a connection string a logical identification. When specifying the connection the logical identification is used rather then the actual connection string. This allows a setup where a program uses the same set of tables in multiple databases to only have one set of intermediate files.

In previous versions if a program needed to access the same set of tables in multiple databases, for every database a set of intermediate files had to be created. This was a strain on maintenance efforts for such environments.

The new setup allows the use of one set of intermediate files. The program creates a DataFlex connection id, say "MyID" and the intermediate files refer to this ID in the SERVER_NAME setting: "**SERVER_NAME DFCONNID=MyID**". The program must make sure the DataFlex connection ID is created before the tables are opened.

If DataFlex connection IDs are used, changing table definitions via Database Builder can become quite complicated. Once changes have been made to the tables in one database, the tables in all other database are out of sync with the intermediate files. It is the programmer's responsibility to handle the update of all tables in all databases. One approach is to have a set of database specific intermediate files to exist while the change is being made, another approach is to create a program that makes the changes and run that against all databases.

This technique can be used to simplify deploying an application. It is the programmer's responsibility to ensure that the table definitions in the different databases actually are the same. If there are tables that do not have the same definition in different databases the behavior of the DataFlex Connection ID feature is undefined. It might work, it might crash, it might do anything in between; it is not defined.

A new function in the cCLIHandler class allows creating a DataFlex connection ID. Multiple connection IDs can be created. The function is called **CreateConnectionID** and gets three arguments"

Argument	Description
String sID	The ID that will be used to identify this connection.
String sConnectionString	The actual connection string
Integer iOptions	This argument indicates if silent login logic (see Silent login) must be applied when logging in to the connection. If the argument has the value 1 silent login is applied, if it is 0 silent login is not applied. If the argument is not passed (optional argument) it is assumed to be 0.

To create a connection ID to connect to the Order database on the local SQL Server using the SQL Server Connectivity Kit, one could use the following code:

```
Procedure CreateMyID
```

```
    Integer iResult
    Handle hoCLI

    Get Create U_cCLIHandler to hoCLI
    If (hoCLI <> 0) Begin
        Set psDriverID Of hoCLI To "MSSQLDRV"
        Get CreateConnectionID of hoCLI "MyID" ;
                "SERVER=(local);Trusted_Connection=yes;DATABASE=Order" ;
            1 to iResult
        If (iResult) ;
            Showln "Something went wrong..."
        Else ;
            Showln "Connection id created"

        Send Destroy of hoCLI
    End
End_Procedure // CreateMyID
```

Read only driver level attributes have been defined to get the number of connection IDs and their settings. These attributes are DF_DRIVER_NUMBER_CONNECTION_IDS, DF_DRIVER_CONNECTION_ID, DF_DRIVER_CONNECTION_ID_STRING, DF_DRIVER_CONNECTION_ID_OPTIONS. To show all existing connection IDs one could use the following procedure:

```
Function DriverIndex String sDriver Returns Integer
    String sCurrentDriver
    Integer iDriver
    Integer iNumDrivers

    Get_Attribute DF_NUMBER_DRIVERS To iNumDrivers
    For iDriver From 1 To iNumDrivers
        Get_Attribute DF_DRIVER_NAME Of iDriver To sCurrentDriver
        If (Uppercase(sDriver) = Uppercase(sCurrentDriver)) ;
            Function_Return iDriver
    Loop

    Function_Return 0
End_function // DriverIndex

Procedure ShowConnectionIDs
    Integer iDriver
    Integer iNumConn
    Integer iConn
    String sID
    String sConnString
    Integer iConnOptions

    Get DriverIndex "MSSQLDRV" To iDriver
    If (iDriver = 0) Begin
        Showln "Driver not loaded"
        Procedure_Return
    End

    Get_Attribute DF_DRIVER_NUMBER_CONNECTION_IDS of iDriver to iNumConn
    Showln "Number of connection ids: " iNumConn
    For iConn From 0 to (iNumConn - 1)
        Get_Attribute DF_DRIVER_CONNECTION_ID          of iDriver iConn to sID
        Get_Attribute DF_DRIVER_CONNECTION_ID_STRING   of iDriver iConn to sConnString
        Get_Attribute DF_DRIVER_CONNECTION_ID_OPTIONS of iDriver iConn to iConnOptions
        Showln sID ", " sConnString ", " iConnOptions
    Loop
End_Procedure
```

Individual connection ID setting cannot be edited. It is however possible to delete any existing connection ID using the **DeleteConnectionID** function. This function has two arguments:

Argument	Description
String sID	The ID of the connection to delete, can be “” if the index is used.
Integer iIndex	The index of the connection id, can be -1 if the ID is used.

To delete the connection identified by the sting “MyID” one could use the following code:

```
Procedure DeleteMyID
    Integer iResult
    Handle hoCLI

    Get Create U_cCLIHandler to hoCLI
    If (hoCLI <> 0) Begin
        Set psDriverID Of hoCLI To "MSSQLDRV"
        Get DeleteConnectionID of hoCLI "MYID" -1 to iResult
        If (iResult) ;
            Showln "Something went wrong..."
        Else ;
            Showln "Connection deleted"

        Send Destroy of hoCLI
    End
End_Procedure // DeleteMyID
```

In order to be able to use tools for tables that make use of DataFlex Connection IDs and as a possible way to create connection IDs for end users connection IDs can be created in the driver configuration file. This can be done using the **DFCONNECTIONID** keyword. This keyword must be set to three values separated by two comma’s (,). If for example we want to setup an ID called MyID to point to the Order database on our local server not using silent login we would add the following line to the driver configuration file:

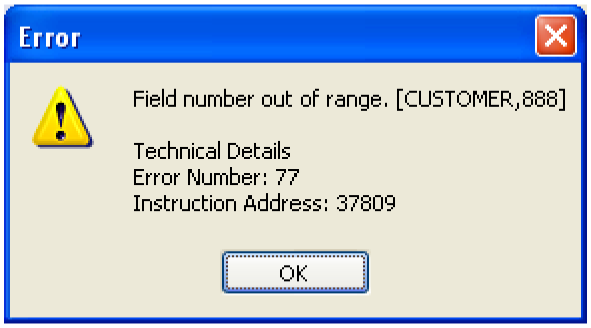
```
DFCONNECTIONID MyID, SERVER=(local); DATABASE=Order, 0
```

Ignore warnings

CLI reports warnings and errors via the same mechanism. The CLI Connectivity Kits report both the warnings and the errors as an error to the runtime environment. The DataFlex environment has no warning concept. The ***Ignore_Warning***s driver configuration file keyword can be used to switch of reporting warnings as errors; they are not reported at all when Ignore_Warnings is on.

Error context information

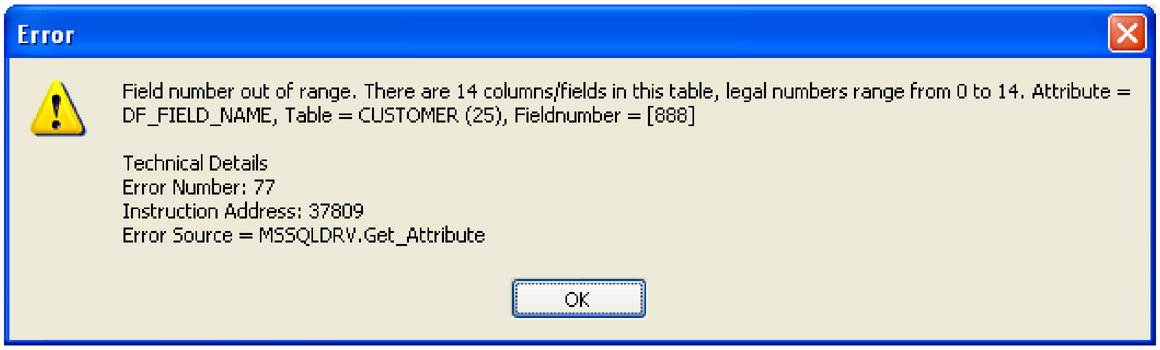
Error context information was added to errors. The context information will contain information that helps resolve the cause of the error. Getting the DF_FIELD_NAME attribute of a non existing field for example; using a previous version of the CK this would result in



```
Field number out of range. [CUSTOMER,888]

Technical Details
Error Number: 77
Instruction Address: 37809
```

now it results in:



```
Field number out of range. There are 14 columns/fields in this table, legal numbers range from 0 to 14. Attribute =
DF_FIELD_NAME, Table = CUSTOMER (25), Fieldnumber = [888]

Technical Details
Error Number: 77
Instruction Address: 37809
Error Source = MSSQLDRV.Get_Attribute
```

Binary column trailing zeroes

Some back-ends will return binary data completely filled out with binary zeroes. If you store hex value "AE003400" in a column of 6 positions you would give "AE0034000000" when retrieving the column.

It is possible to switch on truncation of trailing binary zeroes. This will however also truncate trailing zeroes that are actually part of the column value. Storing the hex value "AE003400" in a column of 6 positions would give "AE0034" when retrieving the column.

You can switch truncating trailing zeroes on by setting the ***Truncate_Binary_Zeroes*** driver configuration keyword to a non zero value.

Embedded SQL – SQLExecDirect

The implementation of the SQLExecDirect Embedded SQL function was cheating. It was not calling the associated CLI function, instead it called SQLPrepare and SQLExecute. This caused a problem in SQL Server. To use the local temporary table feature of SQL Server you must use the SQLExecDirect CLI function. The implementation of our Embedded SQL SQLExecdirect function now calls the correct CLI function.

New attributes

DF_FIELD_NATIVE_SIZE

There was no attribute to return the native size of a column. The DF_FIELD_NATIVE_LENGTH attribute returns the length in the record buffer which does not have to be the same as the native size.

Attributes

In Visual DataFlex 11 a number of database API changes have been made. The biggest change was the possibility to access tables that do not have a "recnum-like" column. Another change was to addition of new attribute levels. Until VDF 11 attributes could be defined on the following levels: Global, Table, Column, Index and Index segment. In VDF 11 and higher, attributes can be defined on the following levels: Global, Driver, Database/Connection, Table, Column, Index and Index segment. For version 5 of the CLI based connectivity kits we start top utilize these new attributes. The attributes can be used to get and/or set configuration information. Previous CLI based connectivity kits had the ***CLI_Get_Driver_Attribute***, ***CLI_Set_Driver_Attribute*** and ***CLI_Get_Database_Attrubut***e commands. These commands are marked **obsolete** in version 5 instead you should use ***Get_Attribute*** and ***Set_Attribute***.

Most of the driver- and database level attributes can be setup via configuration files. The attributes allow a program to adjust a configuration setting temporarily; for the rest of the time the program runs or until the attribute gets another value.

Driver attributes

Some of the driver level attributes are also defined at database level. For these attributes the driver level setting defines the default value. This is used in the ODBC Connectivity Kit where it is possible for each connection to point to a different database system. In such a case it is needed to be able to have settings on database/connection level.

In order to manipulate a driver attribute, the driver must be identified in the Get_Attribute/Set_Attribute command. The database API stores information about database drivers and the connections these drivers have in a layered way. You can enumerate through the loaded drivers and every connection opened by these drivers by using the DF_NUMBER_DRIVERS, DF_DRIVER_NAME, DF_DRIVER_NUMBER_SERVERS, and DF_DRIVER_SERVER_NAME global attributes. IN order to get the index/identifier of a given driver one can use the DriverIndex function below:

```
Function DriverIndex String sDriver Returns Integer
    String sCurrentDriver
    Integer iDriver
    Integer iNumDrivers

    Get_Attribute DF_NUMBER_DRIVERS To iNumDrivers
    For iDriver From 1 To iNumDrivers
        Get_Attribute DF_DRIVER_NAME Of iDriver To sCurrentDriver
        If (Uppercase(sDriver) = Uppercase(sCurrentDriver)) ;
```

```
            Function_Return iDriver
        Loop

        Function_Return 0
    End_function // DriverIndex
```

To get the index/identification for the SQL Server Connectivity Kit for example one would use:

```
Get DriverIndex “MSSQLDRV” To iTheIndex
```

A procedure that shows all driver attributes for the SQL Server Connectivity Kit is listed below:

```
Procedure OnClick
  Integer iDriver
  Integer iAttribValue
  String sAttribValue

  Get DriverIndex "MSSQLDRV" To iDriver
  If (iDriver = 0) Begin
    Showln "Driver not loaded"
    Procedure_Return
  End

  Showln "   Defaults:"
  Get_Attribute DF_DRIVER_DEFAULT_NULLABLE_ASCII Of iDriver To iAttribValue
  Showln "    - nullable Ascii: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DRIVER_DEFAULT_NULLABLE_NUMERIC Of iDriver To iAttribValue
  Showln "    -        Numeric: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DRIVER_DEFAULT_NULLABLE_DATE Of iDriver To iAttribValue
  Showln "    -           Date: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DRIVER_DEFAULT_NULLABLE_TEXT Of iDriver To iAttribValue
  Showln "    -           Text: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DRIVER_DEFAULT_NULLABLE_BINARY Of iDriver To iAttribValue
  Showln "    -         Binary: " (If(iAttribValue, "YES", "NO"))

  Get_Attribute DF_DRIVER_DEFAULT_DEFAULT_ASCII Of iDriver To sAttribValue
  Showln "    -  default Ascii: "sAttribValue
  Get_Attribute DF_DRIVER_DEFAULT_DEFAULT_NUMERIC Of iDriver To sAttribValue
  Showln "    -        Numeric: " sAttribValue
  Get_Attribute DF_DRIVER_DEFAULT_DEFAULT_DATE Of iDriver To sAttribValue
  Showln "    -           Date: " sAttribValue
  Get_Attribute DF_DRIVER_DEFAULT_DEFAULT_TEXT Of iDriver To sAttribValue
  Showln "    -           Text: " sAttribValue
  Get_Attribute DF_DRIVER_DEFAULT_DEFAULT_BINARY Of iDriver To sAttribValue
  Showln "    -         Binary: " sAttribValue

  Get_Attribute DF_DRIVER_MAX_ACTIVE_STATEMENTS Of iDriver To iAttribValue
  Showln "    - Max Active statements: " iAttribValue
  Get_Attribute DF_DRIVER_DRIVER_DECIMAL_SEPARATOR Of iDriver To iAttribValue
  Showln "    - Decimal separator: " iAttribValue ", " (Character(iAttribValue))
  Get_Attribute DF_DRIVER_DRIVER_THOUSANDS_SEPARATOR Of iDriver To iAttribValue
  Showln "    - Thousands separator: " iAttribValue ", " (Character(iAttribValue))
  Get_Attribute DF_DRIVER_DRIVER_DATE_FORMAT Of iDriver To iAttribValue
  Showln "    - Date format: " iAttribValue ", " (If(iAttribValue = DF_DATE_MILITARY, ;
                "MILITARY", If(iAttribValue = DF_DATE_EUROPEAN, "EUROPEAN", "US")))
  Get_Attribute DF_DRIVER_DRIVER_DATE_SEPARATOR Of iDriver To iAttribValue
  Showln "    - Date separator: " iAttribValue ", " (Character(iAttribValue))
  Get_Attribute DF_DRIVER_DUMMY_ZERO_DATE_VALUE Of iDriver To sAttribValue
  Showln "    - Dummy zero date value: " sAttribValue
  Get_Attribute DF_DRIVER_IGNORE_UCASE_SUPPORT Of iDriver To iAttribValue
  Showln "    - Ignore Ucase support: " (If(iAttribValue, "YES", "NO"))

  Get_Attribute DF_DRIVER_IGNORE_WARNINGS Of iDriver To iAttribValue
  Showln "    - Ignore warnings: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DRIVER_USE_DF_LOCKERROR Of iDriver To iAttribValue
  Showln "    - Use DF lock error: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DRIVER_FIND_CACHE_TIMEOUT Of iDriver To iAttribValue
  Showln "    - Find cache timeout (ms): " iAttribValue
  Get_Attribute DF_DRIVER_JIT_TRESHHOLD Of iDriver To iAttribValue
  Showln "    - JIT Treshold (Mb): " iAttribValue
  Get_Attribute DF_DRIVER_TRUNCATE_BINARY_ZEROES Of iDriver To iAttribValue
  Showln "    - Truncate binary zeroes: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DRIVER_ERROR_DEBUG_MODE Of iDriver To iAttribValue
  Showln "    - Error debug mode: " (If(iAttribValue, "YES", "NO"))

  Get_Attribute DF_DRIVER_USE_CACHE Of iDriver To iAttribValue
  Showln "    - Use cache: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DRIVER_USE_CACHE_EXPIRATION Of iDriver To iAttribValue
  Showln "    - Use cache expiration: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DRIVER_REPORT_CACHE_ERRORS Of iDriver To iAttribValue
  Showln "    - Report cache errors: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DRIVER_CACHE_PATH Of iDriver To sAttribValue
  Showln "    - Cache path: " sAttribValue

  Get_Attribute DF_DRIVER_DEFAULT_TABLE_CHARACTER_FORMAT Of iDriver To sAttribValue
  Showln "    - Default table character format: " sAttribValue
  Get_Attribute DF_DRIVER_APPLICATION_CHARACTER_FORMAT Of iDriver To sAttribValue
  Showln "    - Application character format: " sAttribValue

  Get_Attribute DF_DRIVER_DEFAULT_USE_DUMMY_ZERO_DATE Of iDriver To iAttribValue
  Showln "    - Default use dummy zero date: " (If(iAttribValue, "YES", "NO"))

  Get_Attribute DF_DRIVER_LAST_ERROR_TEXT Of iDriver To sAttribValue
  Showln "    - Last error text: " sAttribValue

  Get_Attribute DF_DRIVER_DEFAULT_RECORD_IDENTITY_HIDING Of iDriver To iAttribValue
  Showln "    - Default record identity hiding: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DRIVER_REPORT_ACTIVE_COLUMN_ERRORS Of iDriver To iAttribValue
  Showln "    - Report active column errors: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DRIVER_SILENT_LOGIN Of iDriver To iAttribValue
  Showln "    - Silent login: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DRIVER_DEFAULT_MAP_TO_RECNUM Of iDriver To iAttribValue
  Showln "    - Default map to recnum: " (If(iAttribValue, "YES", "NO"))
End_Procedure // OnClick
```

The driver attributes are documented below, for every attribute the type, allowed access, legal values and corresponding driver configuration keyword are listed. If an attribute cannot be used for certain Connectivity Kits, this is also listed. All attribute for which no special remarks are made can be used in all CLI based Connectivity Kits.

DF_DRIVER_APPLICATION_CHARACTER_FORMAT

Type	String
Access	Read/Write
Values	“OEM”, “Ansi”
Keyword	Application_Character_Format

The character format, OEM or Ansi, of the application in which the Connectivity Kit is loaded. By default this format is OEM (all DataFlex and Visual DataFlex applications use the OEM code page).

DF_DRIVER_CACHE_PATH

Type	String
Access	Read/Write
Values	Any valid path
Keyword	Cache_Path

Path to the directory where the cache files are stored. By default cache files are stored in the same directory as the intermediate file. Cache files will be used if the DF_DRIVER_USE_CACHE attribute is set to true.

DF_DRIVER_CONNECTION_ID

Type	String
Access	Read only
Values	Any string
Keyword	None

The ID of a specified DataFlex connection, see DataFlex Connection IDs for more information.

DF_DRIVER_CONNECTION_ID_STRING

Type	String
Access	Read only
Values	Any connection string for the back end
Keyword	None

The actual connection string of a specified DataFlex connection, see DataFlex Connection IDs for more information.

DF_DRIVER_CONNECTION_ID_OPTIONS

Type	Integer
Access	Read only
Values	0, 1
Keyword	None

The options of a specified DataFlex connection, see DataFlex Connection IDs for more information.

DF_DRIVER_DEFAULT_DEFAULT_ASCII

Type	String
Access	Read/Write

Values	Any valid default value for an ASCII column
Keyword	Default_Default_Ascii

Sets up the default value that will be used when an ASCII column is created. Columns can be created during conversion or within a restructure operation.

DF_DRIVER_DEFAULT_DEFAULT_BINARY

Type	String
Access	Read/Write
Values	Any valid default value for an binary column
Keyword	Default_Default_Binary

Sets up the default value that will be used when a binary column is created. Columns can be created during conversion or within a restructure operation.

DF_DRIVER_DEFAULT_DEFAULT_DATE

Type	String
Access	Read/Write
Values	Any valid default value for an date column
Keyword	Default_Default_Date

Sets up the default value that will be used when a date column is created. Columns can be created during conversion or within a restructure operation.

DF_DRIVER_DEFAULT_DEFAULT_NUMERIC

Type	String
Access	Read/Write
Values	Any valid default value for a bumeric column
Keyword	Default_Default_Numeric

Sets up the default value that will be used when a numeric column is created. Columns can be created during conversion or within a restructure operation.

DF_DRIVER_DEFAULT_DEFAULT_TEXT

Type	String
Access	Read/Write
Values	Any valid default value for a text column
Keyword	Default_Default_Text

Sets up the default value that will be used when a text column is created. Columns can be created during conversion or within a restructure operation.

DF_DRIVER_DEFAULT_MAP_TO_RECNUM

Type	Boolean

Access	Read/Write
Values	True, False
Keyword	Default_Map_To_Recnum

A table can be a recnum or standard table. If a table is opened and its primary index consists of a single numeric column with no decimals, this attribute defines how the table type is reported. By default such a tale will be reported as a recnum table. Setting this attribute to false will cause the table to be reported as a standard table.

DF_DRIVER_DEFAULT_NULLABLE_ASCII

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Default_Default_Nullable_Ascii

Columns can be allowed to contain null values, so called “nullable” columns. This attribute sets up the default “nullability” of ASCII columns. In general we recommend not to allow null values in columns (of any type). See the section on “Null values and defaults” in the help for more information.

DF_DRIVER_DEFAULT_NULLABLE_BINARY

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Default_Default_Nullable_Binary

Columns can be allowed to contain null values, so called “nullable” columns. This attribute sets up the default “nullability” of binary columns. In general we recommend not to allow null values in columns (of any type). See the section on “Null values and defaults” in the help for more information.

DF_DRIVER_DEFAULT_NULLABLE_DATE

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Default_Default_Nullable_Date

Columns can be allowed to contain null values, so called “nullable” columns. This attribute sets up the default “nullability” of date columns. In general we recommend not to allow null values in columns (of any type). See the section on “Null values and defaults” in the help for more information.

DF_DRIVER_DEFAULT_NULLABLE_NUMERIC

Type	Boolean

Access	Read/Write
Values	True, False
Keyword	Default_Default_Nullable_Numeric

Columns can be allowed to contain null values, so called "nullable" columns. This attribute sets up the default "nullability" of numeric columns. In general we recommend not to allow null values in columns (of any type). See the section on "Null values and defaults" in the help for more information.

DF_DRIVER_DEFAULT_NULLABLE_TEXT

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Default_Default_Nullable_Text

Columns can be allowed to contain null values, so called "nullable" columns. This attribute sets up the default "nullability" of text columns. In general we recommend not to allow null values in columns (of any type). See the section on "Null values and defaults" in the help for more information.

DF_DRIVER_DEFAULT_RECORD_IDENTITY_HIDING

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Default_Record_Identity_Hiding

Up to version 3 of the CLI based Connectivity Kits, record identities were not hidden. A column called "DFRECNUM" was added to the end of a table and this column was visible. Setting this attribute to true will hide these columns from the table definition.

DF_DRIVER_DEFAULT_TABLE_CHARACTER_FORMAT

Type	String
Access	Read/Write
Values	"OEM", "Ansi"
Keyword	Default_Table_Character_Format

The default setting for the DF_FILE_TABLE_CHARACTER_FORMAT attribute of a new table created in a structure operation.

DF_DRIVER_DEFAULT_USE_DUMMY_ZERO_DATE

Type	Boolean
Access	Read/Write

Values	True, False
Keyword	Deafult_Use_dummy_Zero_Date

The default setting for the DF_FILE_USE_DUMMY_ZERO_DATE attribute of a new table created in a structure operation.

DF_DRIVER_DRIVER_DATE_FORMAT

Type	Enumerated type
Access	Read/Write
Values	DF_DATE_USA, DF_DATE_EUROPEAN, DF_DATE_MILITARY
Keyword	Driver_Date_Format
Applies to	**ODBC only**

ODBC demands that drivers use the military format (yyy-mm-dd) in the supported SQL. Nevertheless there are drivers that do not use this format but rather a format that depends on the country settings of the machine or the database client software. For those environments the driver date format can be set.

DF_DRIVER_DRIVER_DATE_SEPARATOR

Type	Integer
Access	Read/Write
Values	Ascii value of separator character
Keyword	Driver_Date_Separator
Applies to	**ODBC only**

ODBC demands that drivers use the military format (yyy-mm-dd) in the supported SQL. Nevertheless there are drivers that do not use this format but rather a format that depends on the country settings of the machine or the database client software. For those environments the driver date separator can be set.

DF_DRIVER_DRIVER_DECIMAL_SEPARATOR

Type	Integer
Access	Read/Write
Values	Ascii value of decimal character
Keyword	Driver_Decimal_Separator
Applies to	**ODBC only**

ODBC demands that drivers use the US decimal separator in the supported SQL. Nevertheless there are drivers that do not use this separator but rather a separator that depends on the country settings of the machine or the database client software. For those environments the driver separators can be set.

DF_DRIVER_DRIVER_THOUSANDS_SEPARATOR

Type	Integer
Access	Read/Write
Values	Ascii value of thousands character
Keyword	Driver_Thousands_Separator
Applies to	**ODBC only**

ODBC demands that drivers use the US thousands separator in the supported SQL. Nevertheless there are drivers that do not use this separator but rather a separator that depends on the country settings of the machine or the database client software. For those environments the driver separators can be set.

DF_DRIVER_DUMMY_ZERO_DATE_VALUE

Type	String
Access	Read/Write
Values	Any legal date value on the back end
Keyword	Dummy_Zero_Date_Value
Applies to	**ODBC only**

Sets up the value of the dummy zero date. This should be set to the lowest possible date value that the database supports. In most databases the default 0001-01-01 can be used. Some databases however support a different lowest possible date value.

DF_DRIVER_ERROR_DEBUG_MODE

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Error_Debug_Mode

When the error debug mode is on all errors generated by the database backend will be displayed in a message box. This mode can be used in procedural environments where the screen space reserved to show error messages is often too small to show the complete text of the error message.

DF_DRIVER_FIND_CACHE_TIMEOUT

Type	Integer
Access	Read/Write
Values	0 ..
Keyword	Find_Cache_Timeout

Sets up the find cache timeout (in milliseconds). When the time between two find operations using the same cache is larger then the timeout, the CK will fetch data instead of reading from the cache. See Block cursors for more information.

DF_DRIVER_IGNORE_UCASE_SUPPORT

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Ignore_Ucase_Support

The Embedded Database support case insensitive index segments. This is a feature that other databases do not support. Some databases support case insensitive columns (by using a case insensitive collating sequence for a column). When converting a table that uses case insensitive index segments, extra columns will be created that store the uppercased value of the original column. This extra column is used in index definitions as a segment to achieve the case insensitive index segment functionality. In order to be able to generate the correct find statements the CK uses the back end's uppercase scalar function in the where clause of the select statement generated to implement a find.

The DF_DRIVER_IGNORE_UCASE_SUPPORT will switch the case insensitive index segment logic on or off. When ignored (true) it is switched off, when not ignored (false) it is switched on.

DF_DRIVER_IGNORE_WARNINGS

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Ignore_Warnings

CLI reports warnings and errors via the same mechanism. The CLI Connectivity Kits report both the warnings and the errors as an error to the runtime environment. The DataFlex environment has no warning concept. The DF_DRIVER_IGNORE_WARNINGS driver attribute can be used to switch of reporting warnings as errors; they are not reported at all when set to true.

See also Ignore warnings.

DF_DRIVER_JIT_TRESHHOLD

Type	Integer
Access	Read/Write
Values	0 ..
Keyword	JIT_Treshhold

The threshold, in Mb, that makes a column to be eligible for JIT binding (default 10 Mb). See JIT Binding for more information on JIT Binding.

DF_DRIVER_LAST_ERROR_TEXT

Type	String

Access	Read Only
Values	The text of the last error generated by the CK
Keyword	None

Whenever an error is generated by the Connectivity Kit, this attribute is updated to reflect the error text.

DF_DRIVER_MAX_ACTIVE_STATEMENTS

Type	Integer
Access	Read/Write
Values	0 ..
Keyword	Max_Active_Satements
Applies to	**ODBC only**

The maximum number of concurrently active statements allowed per connection, 0 (zero) means there is no limit.

This attribute can be queried through ODBC but we have found that some drivers do not return a reliable value. The value of the attribute is used by the Connectivity Kit to determine the size of the statement pool. The Connectivity Kit keeps track of the statements that are used per connection in a Most Recently Used sorted list. If the maximum number of statements is in use, the least recently used statement will be freed whenever an additional statement is required.

Some ODBC drivers return a value of 1 for this attribute. This will make the Connectivity Kit free and re-allocate statements all the time. For some drivers the value is accurate (MS Jet Engine ODBC Driver e.g.) but for others it is not (Oracle ODBC driver e.g.).

DF_DRIVER_NUMBER_CONNECTION_IDS

Type	Integer
Access	Read only
Values	0 ..
Keyword	None

The number of defined DataFlex Connection IDs, see DataFlex Connection IDs for more information.

DF_DRIVER_REPORT_ACTIVE_COLUMN_ERRORS

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Report_Active_Column_Errors

Switches error reporting on active columns on or off. The reporting is off if the attribute is set to 0 (zero), all other integer values will switch reporting on. By default, active column errors are not reported.

DF_DRIVER_REPORT_CACHE_ERRORS

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Report_Cache_Errors

Switches reporting on cache read errors on or off. Cache files will be used if the DF_DRIVER_USE_CACHE attribute is set to true.

DF_DRIVER_SILENT_LOGIN

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Silent_Login

The login logic used a function that caused the database client software to popup a login panel if the information supplied was incomplete or wrong. The pop up can be very convenient because it allows the information in the server identification string to be minimal, individual users can complete the information as they login. The panel makes the login an interactive process. There are situations where this is not desired behavior.

In the driver attribute DF_DRIVER_SILENT_LOGIN can be set to true to switch off the popup panel. If Silent_Login is on and the login information in incomplete or wrong, a login error will be generated and no panel will pop up.

DF_DRIVER_TRUNCATE_BINARY_ZEROES

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Truncate_Binary_Zeroes

Some back-ends will return binary data completely filled out with binary zeroes. If you store hex value “AE003400” in a column of 6 positions you would give “AE0034000000” when retrieving the column.

It is possible to switch on truncation of trailing binary zeroes. This will however also truncate trailing zeroes that are actually part of the column value. Storing the hex value “AE003400” in a column of 6 positions would give “AE0034” when retrieving the column.

You can switch truncating trailing zeroes on by setting the DF_DRIVER_TRUNCATE_BINARY_ZEROES attribute to true.

DF_DRIVER_USE_CACHE

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Use_Cache

Switches the use of structure caching on or off.

DF_DRIVER_USE_CACHE_EXPIRATION

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Use_Cache_Expiration

Switches the structure caching intermediate file expiration checking on or off.

DF_DRIVER_USE_DF_LOCKERROR

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Use_DF_Lockerror

Indicates what error will be generated in case of a deadlock/lock timeout error. If true, DFERR_LOCK_TIMEOUT will be generated. This will trigger the build in automatic retry mechanism. If set to false, CLIERR_DEADLOCK_OR_TIMEOUT will be generated.

Database Attributes

In a CLI based connectivity kit the first step is to establish a connection. It is possible to have more then one connection open simultaneously. On connection level a number of attributes have been defined. These attributes can differ in value per connection.

Some of the database level attributes are also defined at driver level. For these attributes the driver level setting defines the default value. This is used in the ODBC Connectivity Kit where it is possible for each connection to point to a different database system. In such a case it is needed to be able to have settings on database/connection level.

Database level attributes can also be setup via a database configuration file. This file is only used by the ODBC Connectivity Kit. The SQL Server and DB2 Connectivity Kit only support a driver configuration file. The Connectivity Kit will copy the appropriate driver level settings to the database level when logging into the database. After this copy operation the ODBC Connectivity Kit will read the database configuration file. The SQL Server and DB2 Connectivity Kit will only copy the settings.

In order to manipulate a database attribute, the driver and database must be identified in the Get_Attribute/Set_Attribute command. The database API stores information about database drivers and the connections these drivers have in a layered way. You can enumerate through the loaded drivers and every connection opened by these drivers by using the DF_NUMBER_DRIVERS, DF_DRIVER_NAME, DF_DRIVER_NUMBER_SERVERS, and DF_DRIVER_SERVER_NAME global attributes. In order to get the index/identifier of a given driver one can use the DriverIndex function described earlier in the section on driver attributes. See Driver attributes. To show a list of all connections and the configuration setting of each connection used by the SQL Server Connectivity Kit use the code below:

```
Procedure OnClick
  Integer iNumDrivers
  String sDriver
  Integer iDriver
  Integer iNumServers
  String sServer
  Integer iServer
  Handle hDatabase

  Get DriverIndex "MSSQLDRV" To iDriver
  If (iDriver <> 0) Begin
    Showln "SQL Server driver is located at: " iDriver
    Showln " Databases:"
    Get_Attribute DF_DRIVER_NUMBER_SERVERS Of iDriver To iNumServers
    For iServer From 1 To iNumServers
      Get_Attribute DF_DRIVER_SERVER_NAME Of iDriver iServer To sServer
      Showln " - " sServer
      Get_Attribute DF_DATABASE_ID Of iDriver iServer To hDatabase
      Showln "Database handle: " hDatabase
      Send ShowDBConfig iDriver hDatabase
    Loop
  End
  Showln
  Showln "Done..."
End_Procedure // OnClick

Procedure ShowDBConfig Integer iDriver Handle hDatabase
  Integer iAttribValue
  String sAttribValue
  Integer iNumAttributes
  Integer iAttribidx

  Showln "   Defaults:"
  Get_Attribute DF_DATABASE_DEFAULT_NULLABLE_ASCII Of iDriver hDatabase To iAttribValue
  Showln "    - nullable Ascii: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DATABASE_DEFAULT_NULLABLE_NUMERIC Of iDriver hDatabase To
iAttribValue
  Showln "    -       Numeric: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DATABASE_DEFAULT_NULLABLE_DATE Of iDriver hDatabase To iAttribValue
  Showln "    -          Date: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DATABASE_DEFAULT_NULLABLE_TEXT Of iDriver hDatabase To iAttribValue
  Showln "    -          Text: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DATABASE_DEFAULT_NULLABLE_BINARY Of iDriver hDatabase To iAttribValue
  Showln "    -        Binary: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DATABASE_DEFAULT_DEFAULT_ASCII Of iDriver hDatabase To sAttribValue
  Showln "    -  default Ascii: "sAttribValue
  Get_Attribute DF_DATABASE_DEFAULT_DEFAULT_NUMERIC Of iDriver hDatabase To sAttribValue
  Showln "    -        Numeric: " sAttribValue
  Get_Attribute DF_DATABASE_DEFAULT_DEFAULT_DATE Of iDriver hDatabase To sAttribValue
  Showln "    -           Date: " sAttribValue
  Get_Attribute DF_DATABASE_DEFAULT_DEFAULT_TEXT Of iDriver hDatabase To sAttribValue
  Showln "    -           Text: " sAttribValue
  Get_Attribute DF_DATABASE_DEFAULT_DEFAULT_BINARY Of iDriver hDatabase To sAttribValue
  Showln "    -         Binary: " sAttribValue

  Get_Attribute DF_DATABASE_MAX_ACTIVE_STATEMENTS Of iDriver hDatabase To iAttribValue
  Showln "    - Max Active statements: " iAttribValue
  Get_Attribute DF_DATABASE_DRIVER_DECIMAL_SEPARATOR Of iDriver hDatabase To iAttribValue
  Showln "    - Decimal separator: " iAttribValue ", " (Character(iAttribValue))
  Get_Attribute DF_DATABASE_DRIVER_THOUSANDS_SEPARATOR Of iDriver hDatabase To
iAttribValue
  Showln "    - Thousands separator: " iAttribValue ", " (Character(iAttribValue))
  Get_Attribute DF_DATABASE_DRIVER_DATE_FORMAT Of iDriver hDatabase To iAttribValue
  Showln "    - Date format: " iAttribValue ", " (If(iAttribValue = DF_DATE_MILITARY, "MILITARY", ;
            If(iAttribValue = DF_DATE_EUROPEAN, "EUROPEAN", "US")))
  Get_Attribute DF_DATABASE_DRIVER_DATE_SEPARATOR Of iDriver hDatabase To iAttribValue
  Showln "    - Date separator: " iAttribValue ", " (Character(iAttribValue))
  Get_Attribute DF_DATABASE_DUMMY_ZERO_DATE_VALUE Of iDriver hDatabase To sAttribValue
  Showln "    - Dummy zero date value: " sAttribValue
  Get_Attribute DF_DATABASE_IGNORE_UCASE_SUPPORT Of iDriver hDatabase To iAttribValue
  Showln "    - Ignore Ucase support: " (If(iAttribValue, "YES", "NO"))

  Get_Attribute DF_DATABASE_INDEX_CREATE Of iDriver hDatabase To iAttribValue
  Showln "    - Index create: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DATABASE_INDEX_DROP Of iDriver hDatabase To iAttribValue
  Showln "    - Index drop: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DATABASE_INDEX_ASC Of iDriver hDatabase To iAttribValue
  Showln "    - Index asc: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DATABASE_INDEX_DESC Of iDriver hDatabase To iAttribValue
  Showln "    - Index desc: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DATABASE_COLUMN_CREATE_DEFAULTCLAUSE Of iDriver hDatabase To
iAttribValue
  Showln "    - Column create default: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DATABASE_DUPREC_STATE Of iDriver hDatabase To sAttribValue
  Showln "    - Duplicate record state: " sAttribValue
  Get_Attribute DF_DATABASE_DUPREC_ERRORNUMBER Of iDriver hDatabase To iAttribValue
  Showln "    - Duplicate record errornumber: " iAttribValue
  Get_Attribute DF_DATABASE_USE_IDENTITY_TYPE Of iDriver hDatabase To iAttribValue
  Showln "    - Use identity type: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DATABASE_MYSQLTABLETYPE Of iDriver hDatabase To sAttribValue
  Showln "    - MySQL Table type: " sAttribValue
  Get_Attribute DF_DATABASE_LOCK_STATE Of iDriver hDatabase To sAttribValue
  Showln "    - Lock state: " sAttribValue
  Get_Attribute DF_DATABASE_NUMBER_NATIVE_LOCKERRORS Of iDriver hDatabase To
iNumAttributes
  Showln "    - Number of lock errors: " iNumAttributes
  For iAttribIdx From 0 To (iNumAttributes - 1)
      Get_Attribute DF_DATABASE_NATIVE_LOCKERROR Of iDriver hDatabase iAttribIdx To
iAttribValue
      Showln "    - Lock error: " iAttribValue
  Loop
  Get_Attribute DF_DATABASE_DEFAULT_MAX_ROWS Of iDriver hDatabase To iAttribValue
  Showln "    - Default max rows: " iAttribValue

  Get_Attribute DF_DATABASE_IGNORE_WARNINGS Of iDriver hDatabase To iAttribValue
  Showln "    - Ignore warnings: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DATABASE_USE_DF_LOCKERROR Of iDriver hDatabase To iAttribValue
  Showln "    - Use DF lock error: " (If(iAttribValue, "YES", "NO"))
  Get_Attribute DF_DATABASE_FIND_CACHE_TIMEOUT Of iDriver hDatabase To iAttribValue
  Showln "    - Find cache timeout (ms): " iAttribValue
  Get_Attribute DF_DATABASE_JIT_TRESHHOLD Of iDriver hDatabase To iAttribValue
  Showln "    - JIT Treshold (Mb): " iAttribValue
  Get_Attribute DF_DATABASE_TRUNCATE_BINARY_ZEROES Of iDriver hDatabase To iAttribValue
  Showln "    - Truncate binary zeroes: " (If(iAttribValue, "YES", "NO"))
End_procedure // ShowConfig
```

The database attributes are documented below, for every attribute the type, allowed access, legal values and corresponding database configuration keyword (if any) are listed. If an attribute cannot be used for certain Connectivity Kits, this is also listed. All attribute for which no special remarks are made can be used in all CLI based Connectivity Kits.

DF_DATABASE_COLUMN_CREATE_DEFAULTCLAUSE

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Column_Create_Defaultclause
Applies to	**ODBC only**

Some ODBC drivers do not accurately report capabilities. A number of settings have been created to correct that problem. If a driver reports that it support a feature it really does not support (or vice versa), errors can occur.

The DF_DATABASE_COLUMN_CREATE_DEFAULTCLAUSE can be used to setup column default support. A column default value is used when creating a new record and no value is supplied for the column. The actual default value can be set in the DF_FIELD_DEFAULT_VALUE attribute.

DF_DATABASE_DEFAULT_DEFAULT_ASCII

Type	String
Access	Read/Write
Values	Any valid default value for an ASCII column
Keyword	Default_Default_Ascii

Sets up the default value that will be used when an ASCII column is created. Columns can be created during conversion or within a restructure operation.

DF_DATABASE_DEFAULT_DEFAULT_BINARY

Type	String
Access	Read/Write
Values	Any valid default value for an binary column

Keyword	Default_Default_Binary

Sets up the default value that will be used when a binary column is created. Columns can be created during conversion or within a restructure operation.

DF_DATABASE_DEFAULT_DEFAULT_DATE

Type	String
Access	Read/Write
Values	Any valid default value for an date column
Keyword	Default_Default_Date

Sets up the default value that will be used when a date column is created. Columns can be created during conversion or within a restructure operation.

DF_DATABASE_DEFAULT_DEFAULT_NUMERIC

Type	String
Access	Read/Write
Values	Any valid default value for a bumeric column
Keyword	Default_Default_Numeric

Sets up the default value that will be used when a numeric column is created. Columns can be created during conversion or within a restructure operation.

DF_DATABASE_DEFAULT_DEFAULT_TEXT

Type	String
Access	Read/Write
Values	Any valid default value for a text column
Keyword	Default_Default_Text

Sets up the default value that will be used when a text column is created. Columns can be created during conversion or within a restructure operation.

DF_DATABASE_DEFAULT_MAX_ROWS

Type	Integer
Access	Read/Write
Values	0..
Keyword	Default_Max_Rows

The default setting for the DF_FILE_MAX_ROWS_FETCHED attribute when creating new tables in a structure operation. For some back-ends restricting the size of the result set increases performance. We found this to be the case for MySQL for example.

DF_DATABASE_DEFAULT_NULLABLE_ASCII

Type	Boolean

Access	Read/Write
Values	True, False
Keyword	Default_Default_Nullable_Ascii

Columns can be allowed to contain null values, so called "nullable" columns. This attribute sets up the default "nullability" of ASCII columns. In general we recommend not to allow null values in columns (of any type). See the section on "Null values and defaults" in the help for more information.

DF_DATABASE_DEFAULT_NULLABLE_BINARY

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Default_Default_Nullable_Binary

Columns can be allowed to contain null values, so called "nullable" columns. This attribute sets up the default "nullability" of binary columns. In general we recommend not to allow null values in columns (of any type). See the section on "Null values and defaults" in the help for more information.

DF_DATABASE_DEFAULT_NULLABLE_DATE

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Default_Default_Nullable_Date

Columns can be allowed to contain null values, so called "nullable" columns. This attribute sets up the default "nullability" of date columns. In general we recommend not to allow null values in columns (of any type). See the section on "Null values and defaults" in the help for more information.

DF_DATABASE_DEFAULT_NULLABLE_NUMERIC

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Default_Default_Nullable_Numeric

Columns can be allowed to contain null values, so called "nullable" columns. This attribute sets up the default "nullability" of numeric columns. In general we recommend not to allow null values in columns (of any type). See the section on "Null values and defaults" in the help for more information.

DF_DATABASE_DEFAULT_NULLABLE_TEXT

Type	Boolean

Access	Read/Write
Values	True, False
Keyword	Default_Default_Nullable_Text

Columns can be allowed to contain null values, so called "nullable" columns. This attribute sets up the default "nullability" of text columns. In general we recommend not to allow null values in columns (of any type). See the section on "Null values and defaults" in the help for more information.

DF_DATABASE_DRIVER_DATE_FORMAT

Type	Enumerated type
Access	Read/Write
Values	DF_DATE_USA, DF_DATE_EUROPEAN, DF_DATE_MILITARY
Keyword	Driver_Date_Format
Applies to	**ODBC only**

ODBC demands that drivers use the military format (yyy-mm-dd) in the supported SQL. Nevertheless there are drivers that do not use this format but rather a format that depends on the country settings of the machine or the database client software. For those environments the driver date format can be set.

DF_DATABASE_DRIVER_DATE_SEPARATOR

Type	Integer
Access	Read/Write
Values	Ascii value of separator character
Keyword	Driver_Date_Separator
Applies to	**ODBC only**

ODBC demands that drivers use the military format (yyy-mm-dd) in the supported SQL. Nevertheless there are drivers that do not use this format but rather a format that depends on the country settings of the machine or the database client software. For those environments the driver date separator can be set.

DF_DATABASE_DRIVER_DECIMAL_SEPARATOR

Type	Integer
Access	Read/Write
Values	Ascii value of decimal character
Keyword	Driver_Decimal_Separator
Applies to	**ODBC only**

ODBC demands that drivers use the US decimal separator in the supported SQL. Nevertheless there are drivers that do not use this separator but rather a separator that depends on the country settings of the machine or the database client software. For those environments the driver separators can be set.

DF_DATABASE_DRIVER_THOUSANDS_SEPARATOR

Type	Integer
Access	Read/Write
Values	Ascii value of thousands character
Keyword	Driver_Thousands_Separator
Applies to	**ODBC only**

ODBC demands that drivers use the US thousands separator in the supported SQL. Nevertheless there are drivers that do not use this separator but rather a separator that depends on the country settings of the machine or the database client software. For those environments the driver separators can be set.

DF_DATABASE_DUMMY_ZERO_DATE_VALUE

Type	String
Access	Read/Write
Values	Any legal date value on the back end
Keyword	Dummy_Zero_Date_Value
Applies to	**ODBC only**

Sets up the value of the dummy zero date. This should be set to the lowest possible date value that the database supports. In most databases the default 0001-01-01 can be used. Some databases however support a different lowest possible date value.

DF_DATABASE_DUPREC_ERRORNUMBER

Type	Integer
Access	Read/Write
Values	Any integer value
Keyword	Duprec_Errornumber
Applies to	**ODBC only**

The errornumber for the back-end's duplicate records error, when this error occurs the Connectivity Kit will translate it to the DataFlex duplicate record error number (DFERR__DUPLICATE_REC, 28).

DF_DATABASE_DUPREC_STATE

Type	String
Access	Read/Write
Values	SQL Status code
Keyword	Duprec_State

Applies to **ODBC only**

The SQL Status code for the back-end's duplicate records error, when this error occurs the Connectivity Kit will translate it to the DataFlex duplicate record error number (DFERR__DUPLICATE_REC, 28).

DF_DATABASE_FIND_CACHE_TIMEOUT

Type	Integer
Access	Read/Write
Values	0 ..
Keyword	Find_Cache_Timeout

Sets up the find cache timeout (in milliseconds). When the time between two find operations using the same cache is larger then the timeout, the CK will fetch data instead of reading from the cache. See Block cursors for more information.

DF_DATABASE_ID

Type	Handle
Access	Read only
Values	Handle to the database
Keyword	None

The handle to the database connection. This handle serves as the database identification when obtaining other database attributes.

DF_DATABASE_IGNORE_UCASE_SUPPORT

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Ignore_Ucase_Support

The Embedded Database support case insensitive index segments. This is a feature that other databases do not support. Some databases support case insensitive columns (by using a case insensitive collating sequence for a column). When converting a table that uses case insensitive index segments, extra columns will be created that store the uppercased value of the original column. This extra column is used in index definitions as a segment to achieve the case insensitive index segment functionality. In order to be able to generate the correct find statements the CK uses the back end's uppercase scalar function in the where clause of the select statement generated to implement a find.

The DF_DRIVER_IGNORE_UCASE_SUPPORT will switch the case insensitive index segment logic on or off. When ignored (true) it is switched off, when not ignored (false) it is switched on.

DF_DATABASE_IGNORE_WARNINGS

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Ignore_Warnings

CLI reports warnings and errors via the same mechanism. The CLI Connectivity Kits report both the warnings and the errors as an error to the runtime environment. The DataFlex environment has no warning concept. The DF_DRIVER_IGNORE_WARNINGS driver attribute can be used to switch of reporting warnings as errors; they are not reported at all when set to true.

See also Ignore warnings.

DF_DATABASE_INDEX_ASC

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Index_Asc

Some ODBC drivers do not accurately report capabilities. A number of settings have been created to correct that problem. If a driver reports that it support a feature it really does not support (or vice versa), errors can occur.

The DF_DATABASE_INDEX_ASC attribute can be used to setup support for the Asc/Ascending keyword when creating an index. The actual segment direction can be set using the DF_INDEX_SEGMENT_DIRECTION attribute.

DF_DATABASE_INDEX_CREATE

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Index_Create

Some ODBC drivers do not accurately report capabilities. A number of settings have been created to correct that problem. If a driver reports that it support a feature it really does not support (or vice versa), errors can occur.

The DF_DATABASE_INDEX_CREATE attribute can be used to setup support for creating an index. The actual index can be created using the Create_Index command.

DF_DATABASE_INDEX_DESC

Type	Boolean
Access	Read/Write

Values	True, False
Keyword	Index_Desc

Some ODBC drivers do not accurately report capabilities. A number of settings have been created to correct that problem. If a driver reports that it support a feature it really does not support (or vice versa), errors can occur.

The DF_DATABASE_INDEX_DESC attribute can be used to setup support for the Desc/Desscending keyword when creating an index. The actual segment direction can be set using the DF_INDEX_SEGMENT_DIRECTION attribute.

DF_DATABASE_INDEX_DROP

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Index_Drop

Some ODBC drivers do not accurately report capabilities. A number of settings have been created to correct that problem. If a driver reports that it support a feature it really does not support (or vice versa), errors can occur.

The DF_DATABASE_INDEX_DROP attribute can be used to setup support for dropping (deleting) an index. The actual removal of the index can be done using the Delete_Index command.

DF_DATABASE_JIT_TRESHHOLD

Type	Integer
Access	Read/Write
Values	0 ..
Keyword	JIT_Treshhold

The threshold, in Mb, that makes a column to be eligible for JIT binding (default 10 Mb). See JIT Binding for more information on JIT Binding.

DF_DATABASE_LOCK_STATE

Type	String
Access	Read/Write
Values	SQL status codes separated by |
Keyword	Lock_State

The SQL Status codes that are generated when a deadlock or lock timeout error occurs. When an error occurs with a status code in this list the error will be translated to the DataFlex lock timeout error, thus enabling the automatic retry mechanism.

The string can hold multiple SQL States, they just be separated by the '|' character. If a back end generates SQL Status XY0001 and DF0002 in case of a deadlock and lock timeout the DF_DATABASE_LOCK_STATE should be set to "|XY0001|DF0002|".

DF_DATABASE_MAX_ACTIVE_STATEMENTS

Type	Integer
Access	Read/Write
Values	0 ..
Keyword	Max_Active_Satements
Applies to	**ODBC only**

The maximum number of concurrently active statements allowed per connection, 0 (zero) means there is no limit.

This attribute can be queried through ODBC but we have found that some drivers do not return a reliable value. The value of the attribute is used by the Connectivity Kit to determine the size of the statement pool. The Connectivity Kit keeps track of the statements that are used per connection in a Most Recently Used sorted list. If the maximum number of statements is in use, the least recently used statement will be freed whenever an additional statement is required.

Some ODBC drivers return a value of 1 for this attribute. This will make the Connectivity Kit free and re-allocate statements all the time. For some drivers the value is accurate (MS Jet Engine ODBC Driver e.g.) but for others it is not (Oracle ODBC driver e.g.).

DF_DATABASE_MYSQLTABLETYPE

Type	String
Access	Read/Write
Values	"BDB", "HEAP", "InnoDB", "MERGE", "MRG_MYISAM", "MYISAM"
Keyword	Lock_State
Applies to	**ODBC Connected to MySQL only**

The type of the MySQL table that will be created by the ODBC Connectivity Kit. MySQL supports several different table types each having its advantages and disadvantages. See the MySQL documentation for more information on table types. It is recommended to use a table type that supports transactions.

If you get this attribute the value will have a " TYPE = " prefix compared to the original setting; setting the attribute to "InnoDB" will result in " TYPE = InnoDB" when getting it.

DF_DATABASE_NATIVE_LOCKERROR

Type	Integer
Access	Read/Write

Values	Native errornumber
Keyword	Native_Lockerror

The native error numbers that are generated when a deadlock or lock timeout error occurs. When an error occurs with an error number in this list the error will be translated to the DataFlex lock timeout error, thus enabling the automatic retry mechanism.

It is possible to setup multiple error numbers. Every Set_Attribute command of this attribute will add the error number to the already existing list (if any). The list can be emptied out by setting the DF_DATABASE_NUMBER_NATIVE_LOCKERRORS attribute to 0 (zero).

DF_DATABASE_NUMBER_NATIVE_LOCKERRORS

Type	Integer
Access	Read/Write
Values	0..
Keyword	None

The number of native error numbers that are generated when a deadlock or lock timeout error occurs. When an error occurs with an error number in this list the error will be translated to the DataFlex lock timeout error, thus enabling the automatic retry mechanism.

Error number are added to the list by setting the DF_DATABASE_NATIVE_LOCKERROR attribute. The list can be emptied out by setting the DF_DATABASE_NUMBER_NATIVE_LOCKERRORS attribute to 0 (zero).

DF_DATABASE_NUMBER_TYPES

Type	Integer
Access	Read only
Values	0..
Keyword	None

The number of types in the database.

DF_DATABASE_TRUNCATE_BINARY_ZEROES

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Truncate_Binary_Zeroes

Some back-ends will return binary data completely filled out with binary zeroes. If you store hex value "AE003400" in a column of 6 positions you would give "AE0034000000" when retrieving the column.

It is possible to switch on truncation of trailing binary zeroes. This will however also truncate trailing zeroes that are actually part of the column value. Storing the hex value “AE003400” in a column of 6 positions would give “AE0034” when retrieving the column.

You can switch truncating trailing zeroes on by setting the DF_DATABASE_TRUNCATE_BINARY_ZEROES attribute to true.

DF_DATABASE_TYPE_AUTOINC

Type	Boolean
Access	Read only
Values	True, False
Keyword	None

Indicates if the type is an “auto-incement” type.

Several type attributes can be queried. Types are identified by the database handle and the ordinal position of the type.

DF_DATABASE_TYPE_CREATE_PARAMS

Type	String
Access	Read only
Values	“length”, “precision”, “scale”, “”
Keyword	Type_Createparams_L, Type_Createparams_LP, Type_Createparams_NONE

A list of keywords, separated by commas, corresponding to each parameter that the application may specify in parentheses when creating a column of the corresponding type. The keywords in the list can be any of the following: length, precision, or scale. They appear in the order that the syntax requires them to be used. For example, CREATE_PARAMS for DECIMAL would be "precision,scale"; CREATE_PARAMS for VARCHAR would equal "length." An empty string is returned if there are no parameters for the data type definition; for example, INTEGER.

Several type attributes can be queried. Types are identified by the database handle and the ordinal position of the type.

DF_DATABASE_TYPE_ID

Type	Enumerated (integer)
Access	Read only
Values	SQL_UNKNOWN_TYPE, SQL_CHAR, SQL_NUMERIC, SQL_DECIMAL, SQL_INTEGER, SQL_SMALLINT, SQL_FLOAT, SQL_REAL, SQL_DOUBLE, SQL_DATETIME, SQL_VARCHAR, SQL_TYPE_DATE, SQL_TYPE_TIME, SQL_TYPE_TIMESTAMP, SQL_DATE, SQL_INTERVAL, SQL_TIME, SQL_TIMESTAMP,

	SQL_LONGVARCHAR, SQL_BINARY, SQL_VARBINARY, SQL_LONGVARBINARY, SQL_BIGINT, SQL_TINYINT, SQL_BIT, SQL_WCHAR, SQL_WVARCHAR, SQL_WLONGVARCHAR, SQL_GUID.
Keyword	None

The ID of the type.

Several type attributes can be queried. Types are identified by the database handle and the ordinal position of the type.

DF_DATABASE_TYPE_MAXSIZE

Type	Integer
Access	Read only
Values	0..
Keyword	None

The maximum size of the type.

Several type attributes can be queried. Types are identified by the database handle and the ordinal position of the type.

DF_DATABASE_TYPE_NAME

Type	String
Access	Read only
Values	A type name
Keyword	None

The name of the type.

Several type attributes can be queried. Types are identified by the database handle and the ordinal position of the type.

DF_DATABASE_TYPE_UNSIGNED

Type	Boolean
Access	Read only
Values	True, False
Keyword	None

Indicates if the type is unsigned (true) or not (false).

Several type attributes can be queried. Types are identified by the database handle and the ordinal position of the type.

DF_DATABASE_USE_DF_LOCKERROR

Type	Boolean
Access	Read/Write

Values	True, False
Keyword	Use_DF_Lockerror

Indicates what error will be generated in case of a deadlock/lock timeout error. If true, DFERR_LOCK_TIMEOUT will be generated. This will trigger the build in automatic retry mechanism. If set to false, CLIERR_DEADLOCK_OR_TIMEOUT will be generated.

DF_DATABASE_USE_IDENTITY_TYPE

Type	Boolean
Access	Read/Write
Values	True, False
Keyword	Use_Identity_Type
Applies to	**ODBC only**

Indicates if the identity type must be used for the recnum column when creating a recnum table. The identity type is the type that has the auto increment attribute set to true and its ID to SQL_INTEGER. If such a type does not exist, it is the first type that has its auto increment attribute set to true.

Bugs & Features

The first list contains the items that have changed since the first public beta (5.0.0.15), the second list contains the items that have changed between version 4.1.0.0 and 5.0.0.15.

Bug/Feature	Version
ALL: When using embedded SQL with SQLBindfile on a recnum table, the recnum value would not be returned in the buffer.	5.1.0.101
ODBC_DRV: During restructure of a MySQL recnum table the auto_increment setting of the recnum column would be lost. Made a change to preserve the auto_increment on MySQL recnum tables during restructure.	5.1.0.100
MSSQLDRV: Added support for SQL Server time columns.	5.1.0.99
ODBC_DRV: Made changes to better support MySQL 5.5. The handling of MySQLTableType setting/attribute has been changed: MySQLTableType is a clause that must be added to the MySQL Create Table statement This used to be TYPE = <typename> in earlier MySQL versions, but in MySQL 5.5, it can only be ENGINE = <typename> Also the possible type names have changed. Dependent on MySQL version some types no longer exist and new ones have been added. Earlier versions of the ODBC Connectivity Kit had only a fixed list of types to choose from. As of build 5.1.0.98 the setup has been made more flexible.	5.1.0.98

The MySQLTableType clause should be specified in the MySQL.int configuration file, or with the DF_DATABASE_MYSQLTABLETYPE attribute. The full clause, as it will be added to the Create Table statement should be specified. MySQL.int file example: MySQLTableType ENGINE = InnoDB (add no quotes, spaces are allowed) Set Attribute example: Set_Attribute DF_DATABASE_MYSQLTABLETYPE of iDriverId hDatabase to "ENGINE = InnoDB" Default setting: ENGINE = InnoDB Specifying an empty string will turn the setting off. The fixed named types can still be used for backward compatibility reasons. Changed MySQL.int file. • Changed comments/example for the MYSQLTABLETYPE keyword. • Changed the settings for DF_Text and DF_Binary columns to nullable with no default, since MySQL does not allow defaults on BLOB and TEXT columns.	
MSSQLDRV: When using a trigger on a recnum table, in some situations an error on GetScopeIdentity.Fetch could be generated. This would happen if the trigger code generated additional result messages (x row(s) affected).	5.1.0.97
MSSQLDRV: The function EnumerateServers in mssqldrv.pkg always used the SQL Server 2000 client odbc driver. This has now been changed to use the 'highest' installed client odbc driver. Changes were made in mssqldrv.pkg and cli.pkg.	5.1.0.96
ALL: The statement Move (NullDateTime()) to DatetimeColumn (where DatetimeColumn is a nullable column), would generate an "Invalid character value for cast specification" error. This has now been fixed. The value will be stored as a NULL value.	5.1.0.96
ALL: Lifted 'Invalid statement handle' errors after end_transaction or unlock inside embedded SQL loop. Versions of the connectivity kit before 5.1.0.95 would generate 'Invalid statement handle' errors when inside an embedded SQL loop a transaction commit (end_transaction or unlock) or transaction abort (abort_transaction) occurred.	5.1.0.95

This happened because the Connectivity Kit would close all statement handles on all connections (including embedded SQL connections) after a transaction commit. As of build 5.1.0.95 the Connectivity Kit will no longer close the statements on embedded SQL connections that are opened with the SQLConnect function. Statements on shared embedded SQL connections (opened through SQLFileConnect function) will still be closed and will generate 'Invalid statement handle' errors after commit. So if you want to perform transactions (table updates) inside an embedded SQL loop, you must use the SQLConnect function to create the embedded SQL connection.	
ALL: When the length of a binary column was specified in the INT files and it was larger than 32768, the allocated buffer would be too small. This could cause truncated data or cause memory overwrites.	5.1.0.94
ALL: Changed the getrowid function to trim padding spaces from df_ascii fields when creating the rowid. Buffer values for df_ascii fields can sometimes be padded with spaces and sometimes trimmed with no trailing spaces.This would lead to different rowid values. The record would be found by both rowid's, but the IssameRowid function would return false.	5.1.0.94
ODBC_DRV: When connected to a SQLBase database, the SQLBase ODBC driver returned too short lengths for maximum table name length, column name length, schema name length and cursor name length. Created new configuration file keywords to override these maximum lengths: MAX_TABLE_NAME_LEN 30 MAX_COLUMN_NAME_LEN 30 MAX_SCHEMA_NAME_LEN 30 MAX_CURSOR_NAME_LEN 30 These keywords can be specified in a database specific configuration file like SQLBase.int or Oracle.int, etc.	5.1.0.93
cli.pkg: Added SQLSERVER2012CLIENT value for DF_DRIVER_SQLSERVER_CLIENT_VERSION attribute.	5.1.0.93
MSSQLDRV: The MSSQL Connectivity Kit will detect and use the SQL Server 2012 client (SQL Server Native Client 11.0) when it is installed on the client machine.	5.1.0.92
ALL: Find last record on a table with Table_Character_Format set to OEM, would not always find the correct record. This was caused by an incorrect ansi/oem conversion in determining the highest collating value of the backend.	5.1.0.91
ALL: An INT file containing a FIELD_NUMBER larger than the actual number of columns of the backend table, would cause a memory overwrite with unpredictable results. The Connectivity Kit will now raise an error on an invalid FIELD_NUMBER	5.1.0.90

in the INT file.	
ALL: Changed the syntax of the open method introduced in build 5.1.0.88. The open with connect string method now uses the following syntax: <drivername>:[<schemaname>#]<table>@<connectstring>[|options[|<name>= <value>]…] Currently the following |options exist: |intfile=<filepath>.int Example: MSSQLDRV:orderhea@SERVER=MYServer;Trusted_Connection=yes;DATABASE=Orders|options|intfile=Orderhea.int The open string can now be followed by the '|options' keyword and one or more name=value pairs. If the intfile option is not present the table will be opened without an int file. (This is unchanged from earlier builds.) If the intfile option is provided: • The int file will be used to open the table. • The DRIVER_NAME, SERVER_NAME, DATABASE_NAME and SCHEMA_NAME keywords from the INT file will be skipped. Those values will be taken from the open string. • Cache files will not be used. • Restructure of the table is not possible. New error message added (dferr003.dat) DFODBCERR_INVALIDCONNECTSTRINGOPENOPTION /* 12342 */ Invalid connect string open option	5.1.0.89
ALL: Added INT file support for opening files using the direct open method (<CKId>:<Schema>#<Table>@<ConnectString>|<Filename>.int") Now the @<ConnectString> can be followed by a pipe character '|' followed by the name of the INT file. Doing so, the driver will apply the information from the .INT file to the passed direct open string. Note that when an INT file has been passed, it will skip the keyword DRIVER_NAME, SERVER_NAME, DATABASE_NAME and SCHEMA_NAME from the INT file because those are supposed to be supplied in the string already. For example: "MSSQLDRV:dbo#customer@SERVER=.\SQLEXPRESS;Trusted_Connectio	5.1.0.88

n=yes;DATABASE=OrderEntrySQL|CUSTOMER.INT" AS Customer	
ALL: Getting the value from a binary column when using the driver with the DataFlex Console Mode runtime would not return the full length of the binary data.	5.1.0.87
ALL: The way the FILL_FIELD function works has been changed for DATE and DATETIME based columns. When filling the column with DF_LOW, it will put a 0 value in the column instead of the lowest possible date or datetime value for the database. The connectivity kit will take care of converting the 0 value into the lowest possible value when actually accessing the database. This change makes the FILL_FIELD command work consistently across all connectivity kits.	5.1.0.86
ALL: Using SQLBindFile on a table that has the TABLE_CHARACTER_FORMAT set to ANSI would use an OemToAnsi translation too many, causing characters in the higher regions of the ASCII table (>127) to be wrong.	5.1.0.85
ALL: Calling the function GetRowId for exactly the same row could lead to row id's that were not exactly the same. Calling FindByRowID would always return the same row. Comparing the row id's using IsSameRowID could return a false condition.	5.1.0.84
MSSQLDRV: Fixed a bug where converting a table to Microsoft SQL Server _with_ recnum support would assign a different value to a recnum field of the row when there recnums are not numbered sequentially. This results in rows getting assigned a different (too low) recnum value after a 'hole' in the recnum series.	5.1.0.83
MSSQLDRV: The method for determining the actual used default collating sequence of a server has been changed to speed up the login process of the driver.	5.1.0.82
ALL: Functions for CreateConnectionID and DeleteConnectionID have been made case-insensitive.	5.1.0.81
ALL: Redirecting a connection using the function RedirectConnection did not change a connection when the old connection string was completely present in the new connection string.	5.1.0.80

ALL: The function RedirectConnection now supports redirecting to the same connection string. This might seem strange, but it allows one to redefine an existing connection id and then let all tables using that connection id reconnect by executing the following code (assuming your connection id is named MyId):
 	
`Get DeleteConnectionID of hoCLI "MyId" -1 to iRetval`
`Get CreateConnectionID of hoCLI "MyId ;`
`        "SERVER=(local);UID=sa;PWD=sa;DATABASE=MyDb" 1 to iRetval`
`Get RedirectConnection "DFCONNID=MyId" "DFCONNID=MyId" to iRetval`
	5.1.0.80

ALL: An error will be triggered when a connection ID is created that already exists. Error: 12341: DFODBCERR_CONNECTIONIDALREADYEXISTS	5.1.0.80

ALL: Tables opened without INT file that have a primary index on a numeric column were assumed to get a value assigned for the column by the database when inserting new rows. Because of this, tables that have primary index on a numeric column without being 'autoincrement', would not get there value set. This has been changed so that only if the primary index is on a column type that is equal to the database's identity type (e.g. 'int identity'), then it will be assumed that the database automatically assigns a value.	5.1.0.79
ALL: Finding on an empty filebuffer on an index that has uppercase segments could generate error messages. This was introduced in release 5.1.0.73.	5.1.0.78
ALL:The SQLBindFile procedure would not work well if the executed SQL statement would return a mix of columns that could be mapped to a base table and columns that could not be mapped to a base table. More specifically, Microsoft SQL Server would return a base table for unlimited varchar columns, but not for other columns. This has been fixed by ignoring the base table names for unlimited varchar columns.	5.1.0.77
ALL: The driver would truncate data when using embedded SQL and getting the value for a numeric column that was using all positions for that field. For example, for a 4.2 defined field with a value of 1234.56, the driver would return 1234.5. This is fixed. Please note that this also requires a change to the SQL.PKG.	5.1.0.76
MSSQLDRV: Reading newly assigned value for tables with recnum support and which don't have the recnum column defined as the first column of the table now select an index will now select an index, preventing the driver to crash with a null pointer reference.	5.1.0.75
ALL: CCH files are now created by default in the same location where the .INT file is, instead of using the first folder of the DFPATH. The new way is the way it was documented. Use the CACHE_PATH attribute to explicitly define where cache files should be put.	5.1.0.74
ALL: Doing a FIND GE, GT, LE or LT on a clear buffer using an index that contains segments on nullable fields which are neither the first nor the last segment, could make the next segment using the wrong value for seeding the index.	5.1.0.73
ALL: The selection process for mapping a DataFlex field type to a database column type has been altered for numeric fields that do not have decimals. The driver could select a integer size that could not complete fit a fields requested length. For example, a numeric field in DataFlex defined as 10 digits with no decimals, could be assigned to a 32-bit integer. The 32-bit integers can hold 10 digits, but only up to 4294967296 or 2147483648 depending on the sign. The CK will now scale the column type up to a NUMERIC or DECIMAL type.	5.1.0.72
ALL: A clear now initializes a GUID type field to all 0's, otherwise performing a find EQ on cleared buffer that has a GUID in the index would generate a SQL error message about the format of the GUID not being correct.	5.1.0.71
MSSQLDRV: The method for getting an assigned rowid (recnum) after a record has been created has changed from IDENT_CURRENT into SCOPE_IDENTITY because the IDENT_CURRENT could return the assigned identity from a different row that was inserted between inserting the	5.1.0.70

self inserted row and getting the identity for the self inserted row.	
ALL: Getting a binary value now respects the requested length if not all of the value is requested. Requesting less than the actual available data caused a buffer overrun. Problem was exposed by the Visual DataFlex debugger that requests the first 256 bytes of data from a binary column.	5.1.0.69
MSSQLDRV: Finding on index that was based on a "unique identifier" data type would cause an error when the file buffer was empty or contains spaces. Field is now filled with the minimal GUID value.	5.1.0.68
DB2_DRV: Introduced a new driver level attribute SKIP_IDENTITY_COLUMN_CHECK,. When set to true, it will no longer perform a check for the existence of an identity column when a table gets opened. It was reported by Marco Kuipers that this statement would be executed for each open statement. When running in a character mode environment where lots of FLX are 'CHAIN'ed, this may lead to an overhead.	5.1.0.67
DB2_DRV: DF_Binary columns will now be created as "VARCHAR FOR BIT DATA" on DB2. Previously "CHARACTER FOR BIT DATA" was used. With this change DF_Binary columns will no longer be padded with spaces, but with binary zeroes.	5.1.0.66
Find's on index segments with accented characters would sometimes fail if: -Ignore_Ucase_Support true -The table still having uppercased columns.	5.1.0.65
DB2_DRV: The OPTIMIZE FOR 1 ROW option will no longer be added to embedded SQL select statements. Earlier change in build 5.0.0.63 was not correct.	5.0.0.64
Changed default for Ignore_Ucase_Support to true. **Warning: This change may have consequences for existing databases/applications!!** **See "Uppercase Columns" for more information.**	5.0.0.63
The SQLConnect function will now set the transaction isolation level to Read_Uncommited. So far, SQLConnect used the default isolation level of the database backend. For SQL Server and DB2 the default is Read_Committed. For table access the Connectivity Kit uses Read_Uncommitted transaction isolation level. To get the same isolation level for table access and Embedded SQL,	5.0.0.63

SQLConnect will now set the isolation level to Read_Uncommitted.	
DB2_DRV: The OPTIMIZE FOR 1 ROW option will no longer be added to embedded SQL select statements.	5.0.0.63
DB2_DRV for Linux: After restructure with db2_drv for Linux, index.0 would have incorrect name. TABLE0 instead of TABLE000.	5.0.0.63
Support for SQL Server 2008 The SQL Server Connectivity Kit will now use the SQL Server 2008 client ("SQL Server Native Client 10.0") if it is installed on the client machine. The SQL Server Connectivity Kit checks which clients are installed on the client machine and uses the 'highest' available. o "SQL Server Native Client 10.0" (SQL Server 2008 Client) o "SQL Native Client" (SQL Server 2005 Client) o "SQL Server" (SQL Server 2000 Client (MDAC)) The used client version can be queried by the new attribute DF_DRIVER_SQLSERVER_CLIENT_VERSION This attribute can return following values: //*** Possible DF_DRIVER_SQLSERVER_CLIENT_VERSION values #REPLACE SQLSERVERUNKNOWNCLIENT 0 #REPLACE SQLSERVER2000CLIENT 8 // "SQL Server" #REPLACE SQLSERVER2005CLIENT 9 // "SQL Native Client" #REPLACE SQLSERVER2008CLIENT 10 // "SQL Server Native Client 10.0" This attribute cannot be set. The connectivity kit can be configured to raise an error if the client and	5.0.0.62

server version do not match. (See Match_Client_Server_Version)	
The SQL Server Connectivity Kit supports the new date and time data types for SQL Server 2008. SQL Server 2008 introduces new data types for storing date (SQL_DATE) and time (SQL_TIME) variables. SQL server versions before SQL Server 2008 did not have a SQL_Date type, but only a SQL_Datetime type. When creating new columns the Connectivity Kit would map DataFlex DF_Date type columns as SQL_Datetime type. This has been left unchanged. When creating new columns (during conversion or restructure) DF_Date maps to SQL_Datetime DF_Datetime maps to SQL_Datetime When connecting to existing SQL Server tables with the Connect Wizard: SQL_Date maps to DF_Date SQL_Datetime maps to DF_Datetime Note: The new SQL Server 2008 types will only be recognized if the "SQL Server Native Client 10.0" (SQL 2008 client) is used. When using an older client the new types will be returned as SQL_Char/DF_Ascii types.	5.0.0.62
Match_Client_Server_Version (All cli connectivity kits) This setting (MATCH_CLIENT_SERVER_VERSION) can be set in the connectivity kit configuration file (mssqldrv.int, db2_drv.int, odbc_drv.int) or through the DF_DRIVER_MATCH_CLIENT_SERVER_VERSION attribute. If this setting is set to true, the Connectivity Kit will check if the version of the database client and server software match. If the client version is less than the server version, an error will be raised when	5.0.0.62

connecting to the server. If the setting is set to false, versions will not be checked. The default is false.	
When using Embedded SQL (SQLConnect function) the Silent_Login setting was not used.	5.0.0.61
Set_Attribute did not check if the Connectivity Kit had initialized itself. If the first call made to the Connectivity Kit was a Set_Attribute of a DF_DRIVER_ or DF_DATABASE_ attribute, this would cause an access violation.	5.0.0.61
Fixed misspelled attributes in cli.pkg: DF_DRIVER_DEFAULT_TABLE_CHARCATER_FORMAT DF_DRIVER_APPLICATION_CHARCATER_FORMAT	5.0.0.61
SQL Server Connectivity Kit: Descending index segments were not reported as descending when using MS SQL Server 2005 without any Service Pack. This was a side effect of a change in build 5.0.0.53. The Connectivity Kit will now use a workaround when the server version is before SQL Server 2005 SP2.	5.0.0.60
Changed default value for DB2 datetime type to include time portion. Default is now '0001-01-01 00:00:00'	5.0.0.60
The Connectivity Kit for DB2 for Linux raised an error when processing the DFCONNECTIONID keyword in the connectivity kit configuration file (db2_drv.int)	5.0.0.57
Some of the DRVR_* attributes returned incorrect values when used with the Cli_Get_Driver_Attribute command.	5.0.0.56
Changed DF_HIGH character for DB2.	5.0.0.55
Get_Attribute did not check if the Connectivity Kit had initialized itself. If the first call made to the Connectivity Kit was an Get_Attribute, this would cause an access violation.	5.0.0.55
Get_Attribute DF_FIELD_NATIVE_TYPE inside structure_start/structure_end caused unhandled exception.	5.0.0.54

After conversion the ascending/descending flag of index segments was sometimes set incorrect. This could happen if the name of one segment was a substring of another segment. Example: VENDORID (DESC), ID (ASC) In this case the ID segment would be set to descending, since ID is a substring of VENDORID.	5.0.0.53
Added support for setting nullable and default values for DateTime type columns: Added Connectivity Kit configuration file keywords: DEFAULT_NULLABLE_DATETIME DEFAULT_DEFAULT_DATETIME Added new attributes: DF_DATABASE_DEFAULT_DEFAULT_DATETIME DF_DATABASE_DEFAULT_NULLABLE_DATETIME DF_DRIVER_DEFAULT_DEFAULT_DATETIME DF_DRIVER_DEFAULT_NULLABLE_DATETIME Added new attributes to Cli.Pkg Changed Connectivity Kit configuration files MSSQLDRV.INT DB2_DRV.INT ODBC_DRV.INT	5.0.0.53
When changing the column name of a DB2 table in dbBuilder, the column name would be uppercased.	5.0.0.53
When reading the connectivity kit configuration file, a line after a 'line with no spaces' would not be processed. Added new error to error file DFERR003.DAT. DFODBCERR_INVALID_CONFIGURATION_VALUE /*	5.0.0.53

12339 */ Changed Connectivity Kit configuration files: (Added explanation and example for DFCONNECTIONID) MSSQLDRV.INT DB2_DRV.INT ODBC_DRV.INT	
SQL Server Connectivity Kit: If an error occurred during "Copy records on backend" in a restructure operation, the restructure rollback operation that followed would cause a GPF in dbBuilder.	5.0.0.53
After a find on a multisegment index, followed by a restructure, the structure_end would raise a memory overwrite error.	5.0.0.53
When a SQL Filter is active, a Find GE would in some situations not respect the SQL Filter.	5.0.0.53
ODBC Connectivity Kit: When connecting to a FoxPro table a date type field would be mapped to a datetime type field. This caused error: "Bad attribute value. Only columns that have a native datetime type can be set."	5.0.0.53
If the full path name of a cache file (.cch) exceeds 64 characters, the cache file would not be found.	5.0.0.53
Multiple SQLExecute calls generating informational messages would cause a memory violation error.	5.0.0.53
Using an Express Connectivity license and connecting to DB2 Express-C Edition 9.5 or later would create a "Not an Express edition" error. This was caused by a change in the product identifier in DB2 Express-C 9.5.	5.0.0.52
Set beta expiration date for db2_drv for Linux to September 1st 2008.	5.0.0.52
A large DF_OPEN_PATH could cause a memory overwrite. This could sometimes cause a crash in the Studio. This happened when many libraries and/or long workspaces paths were used.	5.0.0.51
ODBC Connectivity Kit: After conversion to Microsoft Access, length/precision of columns with type 'currency' would be incorrect. For example: A DataFlex Num 6.2 would be Num 4.4 on Access after conversion. This was caused by Access always using 4 decimals for currency type.	5.0.0.51
SQL Server Connectivity Kit: When connected to SQL Server 2005, locking did not work properly. In some specific situations this could lead to deadlock	5.0.0.50

errors or other locking related problems.	
Changing the find direction when finding inside a transaction would in some situations find the wrong record. For example: Begin_Transaction Find Ge Find Lt End_Transation Find Lt would not change the find direction and act like a Find Gt.	5.0.0.48
Updating the single record of a system file would create an additional record in the system file in the following situation: Move 1 To Sysfile.Recnum Saverecord Sysfile If Sysfile.Recnum already contained the value 1, the file_status would be set to inactive, causing the save to generate a new record, instead of updating the existing record. This bug was found in DF32 console mode dfBrowse program, where editing a system file record would always result in writing an extra record to the system file.	5.0.0.47
The following code: Clear Sysfile Find Ge Sysfile By Recnum would not find the single record of a system file	5.0.0.47
When getting the value of a datetime field, and using the AM/PM notation for the time part, the returned AM/PM value would sometimes be wrong. This was caused by a CK internal datetime to string conversion, where the AM/PM part would be truncated off the resulting string.	5.0.0.47
Set beta expiration date for db2_drv for linux to June 1st 2008.	5.0.0.46
Build a workaround for Firebird ODBC driver from Easysoft. (Uppercase segment handling)	5.0.0.45
Changed user counting logic for use under Windows Vista.	5.0.0.44
Changed the logic to determine if JIT binding should be used. JIT binding will now only be used if the native length of a column is larger than the JIT threshold.	5.0.0.43
Getting the value of a JIT binded column would fail in the following situation: -Standard table -JIT column (text or binary field) present. -The primary index has one or more uppercased segments.	5.0.0.42

Made some changes to let the odbc connectivity kit function with Firebird database.	5.0.0.41
When using SQL Server 2000 the following errors could sometimes be generated during a save: **HY000 Connection is busy with results for another hstmt** This error is caused by a known bug in SQL Server 2000. A workaround has been implemented to avoid situations where this error can occur.	5.0.0.40
After restructure, the field length for numeric and date fields was sometimes not right. This was caused by not writing FIELD_LENGTH to the <table>.int file.	5.0.0.40
When using embedded SQL the installed MS SQLServer client(s) were not properly determined.	5.0.0.40
Fixed a problem with SQLGetdata. An error was generated when executing SQLGetData on a column of type Varchar(Max). The Varchar(Max) type has an unlimited length. This unlimited length is returned as length = 0 by SQL Server. This could not be handled by the CK.	5.0.0.39
DB2 Connectivity Kit Turned on OPTIMIZEFORNROW option.	5.0.0.38
Added support for CK Bundle license.	5.0.0.37
DB2 Connectivity Kit (both Windows and Linux) . Find First_Record on an index with a date segment would cause a DB2 "invalid datetime syntax" error.	5.0.0.35
A SQL Server 2005 column of type varchar(max), was presented as a DF_ASCII field, instead of a DF_TEXT field. Changed handling of varchar(max) and varchar(binary) types. These types were new in SQL Server 2005	5.0.0.35
Beta expiration date set to December 31st, 2007.	5.0.0.34
When a DF_Collate.Cfg (other than English) was used by the runtime, the highest collating character was not determined correct. This caused a Find Last_Record to find a wrong record (or no record at all)	5.0.0.33
Moving a string with a DateTime value to a DateTime field, for example - Move "31/12/2006 11:11:11.111" to Orderhea.Order_Date - would not save the milliseconds, or would cause "Invalid character value for cast specification" error on save. Type of error depends on the regional setting for the decimal separator and/or date format.	5.0.0.33
After adding a new TEXT field, the length would not be written to the INT file. When opening the table after adding the TEXT field, an incorrect length	5.0.0.33

(the default 16383) would be assumed.	
Opening a system table with a text field caused GPF. This would happen if the text field used JIT binding.	5.0.0.33
The following code: Move "" To Table.Column Save Table If Table.Column uses JIT Binding (e.g. a text field) , and the column already contained data, the empty string would not be saved. The column would keep the old value after a save.	5.0.0.33
A Find Last_Record on an index containing a segment of type date, would give an status 16 error (Please enter a valid date). This happened only on VDF12.0 or earlier runtime.	5.0.0.33
A Find Last_Record on index with last segment is recnum and constrain on first segment would not find a record.	5.0.0.33
When accessing a Pervasive.SQL database through ODBC, deleting or updating existing records would generate errors.	5.0.0.33
Minor changes to Mssqldrv.pkg, Db2_drv.pkg and Odbc_drv.pkg - Some REPLACE lines had a comma at the end. Commas removed. Added comment line "Last updated: May 2, 2007	5.0.0.33
Fixed a bug where converting existing data with null dates to Oracle would fail.	5.0.0.32
Fixed a user counting bug where spaces in the login name would cause errors. Fixed a bug where restructuring an Access recnum table looses the auto number attribute. Setting the dummy update column to a non existent column is now illegal.	5.0.0.31
Added the capability to specify the maximum and minimum date and time values that will be used when seeding a buffer for a find. By default these values are 9999-12-31 23:59:59 and 0001-01-01 00:00:00. You can set each of the individual fields by using one the configuration file keywords listed below. This is for use with ODBC databases, it is not needed in the MSSQL and DB2 drivers.. ;MaxDateYear 9999 ;MinDateYear 1 ;MaxDateMonth 12 ;MinDateMonth 1 ;MaxDateDay 31 ;MinDateDay 1 ;MaxTimeHour 23 ;MinTimeHour 0	5.0.0.29

;MaxTimeMinute 59 ;MinTimeMinute 0 ;MaxTimeSeconds 59 ;MinTimeSeconds 0	
Fixed a bug with Overlap columns and Access.	5.0.0.24
Added support for DateTime columns when used with Visual DataFlex 12.1 or higher. When used with earlier revisions of Visual DataFlex and DataFlex 3.2 Console Mode DateTime columns are still treated like Date columns.	5.0.0.23
Fixed bug when restructuring a standard table that has the Generate_Record_ID_Method set to Identity_Column an error would popup about not being able to locate the temporary table used during restructure and the restructure would fail.	5.0.0.21
Fixed bug in creating error context information on new column. When a column was created with a number that did not exist in the original table and an attribute for the column was set to an illegal value, the application would crash	5.0.0.20
Using new user counting logic. This will synchronize the user count so when a process terminates abnormally it will automatically be removed from the user count list.	5.0.0.19
There was a bug in the transaction rollback logic that would not restore recnum to the value it had when the transaction started. Fixed.	5.0.0.18
Check if SQLFetchScroll is supported if not, use SQLFetch. The new MYOB ODBC driver does not support this function. This resulted in "Function not supported errors" when finding in MYOB tables.	5.0.0.16
Fixed a problem in determining the "unsigndness" of a type. We would conclude a type was signed while it actually was not. This caused conversion errors.	5.0.0.16
When changing a numeric column's definition from having no decimals to having decimals (or vice versa) the type would not be adjusted correctly.	5.0.0.16

The list of fixed bugs since version 4.1.0.0. Not all versions are mentioned in the table below. There have been changes hat are "internal" by nature. Code changes to allow compiling on a different platform, switching tools and such things will not be reported in the list below because they do not fix bugs.

Bug/Feature	**Version**
Fixed a problem in reading the max active statements setting for ODBC.	4.1.0.30
Fixed a problem in open logic with overlap columns	4.1.0.28

Bug/Feature	Version
Fixed a problem when trying to open a table in an OpenRDA database (ODBC)	4.1.0.27
Fixed a problem where a transaction rollback would not restore the original, before transaction, column values.	4.1.0.26
Fixed a number of problems with column of type GUID.	4.1.0.24
Some back ends have hidden/system columns. If such a column is used in an index, the ODBC CK would crash.	4.1.0.22
Some back ends have hidden/system columns. If such a column is defined to be the primary key, the ODBC CK would crash.	4.1.0.21
Remove trailing binary zeroes from binary columns.	4.1.0.20
Added possibility to translate deadlock/lock timeout errors to the DataFlex lock timeout error. This will ensure the automatic retry mechanism is used. Driver configuration keyword UseDFLockError to switch this on/off.	4.1.0.20
Fixed an OEM/Ansi translation problem when table or index name uses diacritical characters.	4.1.0.20
Fixed a problem with reading index definitions of a Foxpro table.	4.1.0.19
Fixed a bug in the dummy update logic that would occur if the 4.1 CK was used in a pre VDF11 runtime.	4.1.0.18
Changed fetch logic to use fetch relative for find by rowid logic.	4.1.0.16
When moving a value to a column that was already there we would sometimes mark the column as changed. This was fixed.	4.1.0.15
Fixed problems when creating recnum tables on MYSQL and trying to use the auto increment logic.	4.1.0.14
Delete would sometimes fail in specific circumstances.	4.1.0.12
Fixed problem in finding in a system table with no indices. Problem would occur when switching from inside to outside of a transaction.	4.1.0.11
We did not set the default for record identity method.	4.1.0.10
Added silent login mode, can be set by using the Silent_Login driver configuration file keyword.	4.1.0.9
Added RedirectConnection logic.	4.1.0.9
Fixed a bug in the logout logic for logging out of a named connection.	4.1.0.9
Fixed a bug that reported a busy cursor when creating records	4.1.0.8
The default type for creating tables was not recnum table. This caused problem in pre VDF11 dbBuilders. When a new table was created it would always be a standard table, the pre-VDF11 runtime cannot handle standard tables.	4.1.0.7

Bug/Feature	**Version**
Trying to convert a table to SQL Server that contains a column with a length of 0 would hang, fixed.	4.1.0.7
The Use_Identiy_Type database configuration file keyword was not respected, fixed.	4.1.0.5
Finding on empty uppercased index segments would go wrong for Oracle, fixed.	4.1.0.5
Introduce a global statement handle for creating new records this speeds up creating multiple records in one table.	4.1.0.4
Fixed an infinite recursion problem when locking in Microsoft Access.	4.1.0.3
Allow full range of negative numbers to be used.	4.1.0.1
Added support for system tables with no index.	4.1.0.1