WHAT'S NEW IN CK VERSION 6?

Last Changed: June 24, 2013.

WHAT'S NEW IN CK VERSION 6

This document describes what is new in version 6 of the CLI based Connectivity Kits compared to version 5.1.

The Data Access CLI based Connectivity Kits are:

- SQL Server Connectivity Kit
- ODBC Connectivity Kit
- DB2 Connectivity Kit for Windows
- DB2 Connectivity Kit for Linux

Major changes:

- Support for all MS SQL Server 2008/2012 types, including the new date, datetime2, and time types
- Support for all IBM DB2 10.1 types
- Configurable mapping from DataFlex types to SQL types
- Improved restructure logic
- Auto-reconnect after lost connection

CHANGES IN CONNECTIVITY KIT TYPE HANDLING

When using Connectivity Kits to access an external database like MS SQL Server or DB2, we have to handle type mappings.

The DataFlex database API knows a fixed number of data types as shown in the following table:

DATAFLEX TYPES

DF_ASCII	**Ascii**
DF_BCD	Numeric
DF_DATE	Date
DF_DATETIME	Datetime
DF_TEXT	Text
DF_BINARY	Binary

SQL based database have their own native types. Some of them are very similar to the DataFlex types, others can be quite different. The number of SQL native types is usually larger than the number of DataFlex types.

When accessing a SQL database from a DataFlex program, the SQL native types must be mapped to one of DataFlex types.

This mapping can take 2 directions:

- Mapping SQL type to DF type
- Mapping DF type to SQL type

MAPPING SQL TYPE TO DF TYPE

SQL to DF type mapping happens when a table is opened from a DataFlex program. The table will request the SQL native type from the database and map it to one of the DataFlex types.

This is also used when connecting to an existing table with the Connect Wizard in VDF studio.

The Connectivity Kit for SQL Server will now recognize all native SQL Server 2008/2012 types on open. This includes the new date and time types introduced in SQL Server 2008. All types will be mapped to an appropriate DataFlex type. Data can be read from and written to all SQL Server types.

The Connectivity Kit for DB2 will now recognize all native DB2 10.1 types on open. All types will be mapped to an appropriate DataFlex type. Data can be read from and written to all SQL Server types.

The following section contains tables that show all SQL to DataFlex type mappings for SQL Server and DB2.

MAPPING DF TYPE TO SQL TYPE

DF to SQL type mapping happens when creating new columns in a table through the Connectivity Kit. This is usually done with Table Editor or Database Builder. When creating new columns in a table, the DataFlex type is entered, and this will be mapped to the SQL native type when the table is created or restructured.

Conversion from DataFlex embedded database to SQL can also be considered a restructure.

Earlier versions of the Connectivity Kit would map each DataFlex type to a fixed SQL native type. With Connectivity Kit version 6, each DataFlex type can be mapped to different SQL types by configuration settings and attributes.

CHANGES IN SQL TO DF MAPPING

SQL to DataFlex type mapping happens when the Connectivity Kit opens a backend table. During the open the SQL backend type of the columns will be mapped to a DF type.

> *Note that the opposite mappings (DF to SQL) happen only during restructure. When creating new columns the type will be determined by these mappings*

The backend table can be a table that is originally created by the CK, but it can be also be an external table not created by the CK. For example created by a SQL backend tool or it could be part of a third party application.

In some situations the DataFlex type can be changed by a setting in the table.int file.

It is also possible to change the length on the DataFlex side by specifying a different length in the int file.

CHANGES IN DF TO SQL MAPPING

Mapping of DF type to SQL type happens when the CK perform a restructure. For example when adding a new column or changing the type of a column. Conversion from DataFlex to SQL can also be considered a restructure.

In CK6 the mapping of a DF type to a SQL type has been made configurable. The mappings can be configured by specifying the mapping in the driver configuration file (mssqldrv.int, db2_drv.int, odbc_drv.int) or through corresponding attributes.

It is also possible to specify a default map schema. A default map schema will define DF to SQL mappings for all types. This can be used for backward compatibility with earlier version of the Connectivity Kit, or to develop for a specific version of SQL Server.

> Note:
> All configurable DF to SQL mappings will only be used when creating new columns through the Connectivity Kit. The mapping configuration settings will have no effect for existing columns.
> Existing columns will keep the SQL type they already have and will not be changed by a restructure.

NEW CONFIGURATION FILE KEYWORDS

The following configuration file keywords have been added. These keywords can be specified in the mssqldrv.int and db2_drv.int configuration file.

```
DEFAULT_MAP_DF_TO_SQL_TYPE_SCHEMA

MAP_DFDATE_TO_SQLTYPE
MAP_DFDATETIME_TO_SQLTYPE
MAP_DFASCII_TO_SQLTYPE
MAP_DFTEXT_TO_SQLTYPE
MAP_DFBINARY_TO_SQLTYPE
```

For the ODBC connectivity kit a different set of configuration keywords has been added. Since the ODBC connectivity kit can connect to different database backends with different native types, an ODBC type must be specified in the odbc_drv.int

```
MAP_DFDATE_TO_ODBCTYPE
MAP_DFDATETIME_TO_ODBCTYPE
MAP_DFASCII_TO_ODBCTYPE
MAP_DFTEXT_TO_ODBCTYPE
MAP_DFBINARY_TO_ODBCTYPE
```

NEW ATTRIBUTES

The DF to SQL mappings can also be specified at runtime with the set_attribute command. The following attributes have been added.

```
DF_DRIVER_DEFAULT_MAP_DF_TO_SQL_TYPE_SCHEMA
DF_DATABASE_DEFAULT_MAP_DF_TO_SQL_TYPE_SCHEMA

DF_DRIVER_MAP_DFDATE_TO_SQLTYPE
DF_DRIVER_MAP_DFDATETIME_TO_SQLTYPE
DF_DRIVER_MAP_DFASCII_TO_SQLTYPE
DF_DRIVER_MAP_DFTEXT_TO_SQLTYPE
DF_DRIVER_MAP_DFBINARY_TO_SQLTYPE

DF_DATABASE_MAP_DFDATE_TO_SQLTYPE
DF_DATABASE_MAP_DFDATETIME_TO_SQLTYPE
DF_DATABASE_MAP_DFASCII_TO_SQLTYPE
DF_DATABASE_MAP_DFTEXT_TO_SQLTYPE
DF_DATABASE_MAP_DFBINARY_TO_SQLTYPE
```

Only for ODBC Connectivity Kit the following attributes have been added:

```
DF_DRIVER_MAP_DFDATE_TO_ODBCTYPE
DF_DRIVER_MAP_DFDATETIME_TO_ODBCTYPE
DF_DRIVER_MAP_DFASCII_TO_ODBCTYPE
DF_DRIVER_MAP_DFTEXT_TO_ODBCTYPE
DF_DRIVER_MAP_DFBINARY_TO_ODBCTYPE

DF_DATABASE_MAP_DFDATE_TO_ODBCTYPE
DF_DATABASE_MAP_DFDATETIME_TO_ODBCTYPE
DF_DATABASE_MAP_DFASCII_TO_ODBCTYPE
DF_DATABASE_MAP_DFTEXT_TO_ODBCTYPE
DF_DATABASE_MAP_DFBINARY_TO_ODBCTYPE
```

CONFIGURING DF TO SQL MAPPINGS

Type mappings can be specified in the driver configuration file (mssqldrv.int, db2_drv.int, odbc_drv.int) or at runtime through attributes:

Example:
The following example shows how to specify to map DF_DATE to SQL_DATE and DF_DATETIME to SQL_DATETIME2 in the MSSQLDRV.INT file.

```
; MAP_DFDATE_TO_SQLTYPE: The SQL Server type when creating new DF_DATE columns
;                       Allowed values:
;                         date
;                         datetime
MAP_DFDATE_TO_SQLTYPE date

; MAP_DFDATETIME_TO_SQLTYPE: The SQL Server type when creating new DF_DATETIME columns
;                       Allowed values:
;                         datetime2
;                         datetime
MAP_DFDATETIME_TO_SQLTYPE datetime2
```

To set the mappings at runtime , the attributes defined in cli.pkg can be used:
Example:

```
Use cli.pkg
Get DriverIndex "MSSQLDRV" to iDriverId
Set_Attribute DF_DRIVER_MAP_DFDATE_TO_SQLTYPE of iDriverId       to 'date'
Set_Attribute DF_DRIVER_MAP_DFDATETIME_TO_SQLTYPE of iDriverId   to 'datetime2'

Open Salsep
Get_Attribute DF_DATABASE_ID of iDriverId iServerId to hDatabase

Set_Attribute DF_DATABASE_MAP_DFDATE_TO_SQLTYPE of iDriverId hDatabase       to 'date'
Set_Attribute DF_DATABASE_MAP_DFDATETIME_TO_SQLTYPE of iDriverId hDatabase   to 'datetime2'
```

USING DEFAULT MAP SCHEMAS

A default map schema will define DF to SQL mappings for all DF types. This can be used for backward compatibility with earlier version of the Connectivity Kit, or to develop for a specific version of SQL Server.

Map schemas can only be used with SQL Server Connectivity Kit.

Map schemas can be specified in the connectivity kit configuration file (mssqldrv.int) with the DEFAULT_MAP_DF_TO_SQL_TYPE_SCHEMA keyword.

Or map schemas can be specified at runtime with the DF_DRIVER_DEFAULT_MAP_DF_TO_SQL_TYPE_SCHEMA or DF_DATABASE_DEFAULT_MAP_DF_TO_SQL_TYPE_SCHEMA attributes.

SQL Server map schemas

For SQL Server the following default map schema's can be used:

MAP_DF_TO_SQL_TYPE_SQL2000

MAP_DF_TO_SQL_TYPE_SQL2005
MAP_DF_TO_SQL_TYPE_SQL2008
MAP_DF_TO_SQL_TYPE_SQL2012

The following table shows the DF to SQL type mapping for the SQL Server map schemas.

	SQL2000	SQL2005	SQL2008	SQL2012
DF_ASCII	Char	Char	Char	Char
DF_DATE	Datetime	Datetime	Date	Date
DF_DATETIME	Datetime	Datetime	Datetime2	Datetime2
DF_TEXT	Text	Varchar(max)	Varchar(max)	Varchar(max)
DF_BINARY	Binary/image	Varbinary(max)	Varbinary(max)	Varbinary(max)

If no default mapping schema is specified, the SQL2000 mapping schema will be used. These mappings were also used with earlier Connectivity Kit versions.

Changing the mappings may have consequences for existing databases and applications. See below at Backward compatibility.

The per SQL Server version mapping schema's can be used to develop for a specific SQL Server version.

Default mappings can be overwritten per type with the DF_TO_SQLTYPE settings.

If no default mapping schema is specified, the SQL2000 mapping schema will be used.

MIGRATING EXISTING DATABASES TO NEW TYPES

After upgrading from CK5 to CK6 you have a choice whether you want to use the new data types in your existing databases.

Note you don't *have* to change the data types. The Connectivity Kit is fully backward compatible and can keep using the older data types.

If you do want to change any types you can change them by setting the DF_FIELD_NATIVE_TYPE attribute

The following code shows an example how to convert the native type for various SQL Server types.

```
Procedure ConvertNativeTypeExample

    Integer iNumColumns iColumn iDFType iNativeType
    Handle  hTable

    String  sColumnName sNativeTypeName

    Open OrderHea
    Move OrderHea.File_Number to hTable

    Structure_Start hTable

        Get_Attribute DF_FILE_NUMBER_FIELDS of hTable to iNumColumns
        For iColumn from 1 to iNumColumns

            Get_Attribute DF_FIELD_NAME of hTable iColumn to sColumnName
            Get_Attribute DF_FIELD_TYPE of hTable iColumn to iDfType

            Get_Attribute DF_FIELD_NATIVE_TYPE of hTable iColumn to iNativeType
            Get_Attribute DF_FIELD_NATIVE_TYPE_Name of hTable iColumn to sNativeTypeName

            If (iDFType = DF_DATE) Begin
                If (iNativeType = SQL_TYPE_TIMESTAMP) Begin
                    // Convert datetime to date
                    Set_Attribute DF_FIELD_NATIVE_TYPE of hTable iColumn to SQL_TYPE_DATE
                End
            End

            If (iDFType = DF_DATETIME) Begin
                If (iNativeType = SQL_TYPE_TIMESTAMP) Begin
                   // Convert datetime to datetime2
                   Set_Attribute DF_FIELD_NATIVE_TYPE of hTable iColumn to SQL_TYPE_TIMESTAMP2
                End
            End

            If (iDFType = DF_ASCII) Begin
                If (iNativeType = SQL_CHAR) Begin
                    // Convert char to varchar
                    Set_Attribute DF_FIELD_NATIVE_TYPE of hTable iColumn to SQL_VARCHAR

                    // Convert char to nchar
                    //Set_Attribute DF_FIELD_NATIVE_TYPE of hTable iColumn to SQL_WCHAR

                    // Convert char to nvarchar
                    //Set_Attribute DF_FIELD_NATIVE_TYPE of hTable iColumn to SQL_WVARCHAR
                End
            End

            If (iDFType = DF_TEXT) Begin
                If (iNativeType = SQL_LONGVARCHAR) Begin
```

```
                    // Convert text to varchar(max)
                    Set_Attribute DF_FIELD_NATIVE_TYPE of hTable iColumn to SQL_VARCHAR

                    // Convert text to nvarchar(max)
                    //Set_Attribute DF_FIELD_NATIVE_TYPE of hTable iColumn to SQL_WVARCHAR
                End
            End

            If (iDFType = DF_BINARY) Begin
                If (iNativeType = SQL_LONGVARBINARY) Begin
                    // Convert image to varbinary(max)
                    Set_Attribute DF_FIELD_NATIVE_TYPE of hTable iColumn to SQL_VARBINARY

                    // Convert image to binary
                    //Set_Attribute DF_FIELD_NATIVE_TYPE of hTable iColumn to SQL_BINARY
                End
            End

        Loop
    Structure_End hTable

End_Procedure
```

RESTRUCTURE CHANGES

The following changes have been made in restructure logic:

- During a restructure the existing mappings will be preserved. The backend type of an existing column will remain the same after a restructure. This is independent on the configured mappings.

- When creating new columns the configured mappings will determine the backend type of the new columns.

- The restructure process will now try to write a minimal INT file. In CK5 there would always be a Field_Length written to the INT file for Numeric, Date, Datetime, text and binary type fields. This will now only be done if necessary.
  Only when the length of a column is different from the backend type, a length will be written.
  Examples:
  - The length of an integer is 9 positions. A numeric 9.0 column mapped to integer, will not have a field length, a numeric 8.0 column mapped to integer, will have a length 8 in the int file.
  - Decimal types will never have a length in the int file.
  - Date and datetime types will never have a length in the int file.
  - Varchar(x) columns will not have lengths in the int file.
  - Varchar(max) columns will have a length in the int file, because this type has no specific length on the backend

AUTORECONNECT

The CLI based Connectivity Kits now have the ability to automatically reconnect after a lost connection.

A lost connection situation happens if for some reason the connection between a workstation and the database server has been lost.

A lost connection can be caused by several reasons:

Network hick ups. This is more likely on WAN connections and Wifi networks.
Sometimes connections are lost after a certain period of inactivity.
The database server can be temporary unavailable because of maintenance or backups.

Earlier versions of the Connectivity Kit did not handle a lost connection situation. After the connection from a VDF program to a database server had been lost, the connectivity kit was unable to re-establish the connection.
This means that after an interrupted connection, even if it was only for a short while, **all** database operations from the program would generate 'connection lost' errors. The only solution was to end the VDF program and restart it.

RECONNECT LOGIC

The Connectivity Kits version 6 will now try to re-connect to the database server when a connection lost error has been received.

- If the reconnect succeeds (the connection is re-established), the operation that generated the lost connection error (for example a Find operation), will be retried.

- If the reconnect does not succeed (unable to connect to the database server) a 'connection lost' error will be generated.

Notes:

After a successful reconnection inside a Find operation, it will appear to the program as if nothing has happened. The program has executed a successful Find operation and is unaware a re-connect has taken place. Note this will only occur if the first re-connect attempt is successful.

If the first reconnect attempt fails, the Find operation will generate a 'connection lost' error. The next Find operation from the program will again try to reconnect.

After receiving a connection lost error, the Connectivity Kit will not try to reconnect when it is inside a transaction. Instead it will generate 'lost connection' error. This will abort and roll back the transaction. The first operation after the transaction will try to reconnect, and if successful, the program can continue.

Re-executing the operation after a successful reconnect will only be done on Find and Open operations.

Save and Delete operations always take place inside a transaction and will not try to reconnected, but generate an error that will cause a transaction abort.

CONFIGURE AUTORECONNECT

Auto-Reconnect can be configured in the connectivity kit configuration file (mssqldrv.int, db2_drv.int or odbc_drv.int) with the following keyword:

AUTO_RECONNECT 1

The default value is 1 (Auto-reconnect on)

CONNECTION LOST ERROR

The Connectivity Kit will generate the following error when the connection to the database was lost:

> 12343 DFODBCERR_DATABASECONNECTIONLOST
> Connection to database lost

The connectivity kit will translate the database native connection lost errors to this DataFlex error code.

For the SQL Server and DB2 Connectivity Kits, the native connection lost errors are pre-configured and don't have to be setup.

The ODBC Connectivity Kit can connect to different database back ends and the connection lost state(s) must be specified in the database specific configuration file like Oracle.int, Mysql.int, Access.int, etc.

The connection lost state can be specified with the CONNECTION_LOST_STATE keyword. If there are multiple connection lost states, the CONNECTION_LOST_STATE keyword can be specified multiple times.

Example (MySQL.INT)

```
; Connection_Lost_State
;   The native error that indicates a lost connection.
;   This error will translate to a DF lost connection error.
Connection_Lost_State 08S01
```

APPENDIX A: SQL SERVER TYPE CHANGES

SQL SERVER 2008/2012 TO DF MAPPINGS

The following tables show how the SQL Server 2008/2012 native types map to a DF type.

All types shown are supported by CK6. The CK is able to store and retrieve from/to all the mentioned data types.

NUMERIC TYPES:

SQL type	DF Type
Tinyint	Num 3.0
Bigint	Num 14.0
Numeric	Num 0.1 – Num 14.8
Decimal	Num 0.1 – Num 14.8
Money	Num 15,4
SmallMoney	Num 6.4
Int	Num 9.0
Smallint	Num 4.0
Float	Num 14.8
Real	Num 14.8
Bit	Num 1.0

DATE AND TIME TYPES

SQL type	DF Type
Datetimeoffset	Datetime(34.0)
Time	Asc (11) – Asc(19)
Date	Date (10.0)
Datetime2	Datetime (23.0) – Datetime(23.7)
Datetime	Date(10) or Datetime (23,3)
Smalldatetime	Datetime (23.0)

Notes:

The time, datetime2 and datetimeoffset types can have a 0 to 7 positions decimal part. (milliseconds, nanoseconds).
The length on the DataFlex side will vary dependent on the number of decimals of the backend type.
Examples:

- Time(0) will map to asc(11)
- Time(7) will map to asc(19)
- Datetime2(0) will map to Datetime(23.0)
- Datetime2(7) will map to Datetime(23.7)

The SQL datetime type will map to either DF_Date or DF_Datetime, depending on the FIELD_TYPE setting in the table.int file.

More on date/datetime handling in SQL Server below in ….

CHARACTER DATA TYPES

SQL type	DF Type
Char(10)	Asc(10)
Char(255)	Asc(255)
Char(256)	Asc (256)
Char(1000)	Asc(1000)
Varchar(10)	Asc(10)
Varchar(255)	Asc(255)
Varchar(256)	Text(256)
Varchar(1000)	Text(1000)
Varchar(Max)	Text(16383)
Text	Text(16383)

UNICODE CHARACTER DATA TYPES

SQL type	DF Type
nchar(10)	Asc(10)
nchar(255)	Asc(255)
nchar(256)	ASC(256)
nchar(1000)	Asc(1000)
nvarchar(10)	Asc(10)
nvarchar(255)	Asc(255)
nvarchar(256)	Text(256)
nvarchar(1000)	Text(1000)
nvarchar(Max)	Text(16383)
nText	Text(16383)

BINARY DATA TYPES

SQL type	DF Type
binary(50)	binary(50)
binary(1000)	binary(1000)
varbinary(50)	binary(50)
varbinary(1000)	binary(1000)
varbinary(max)	binary(16383)

Image	Binary(16383
Timestamp	Binary(8)

OTHER DATA TYPES

SQL type	DF Type
Xml	Text(16383)
Uniqueidentifier	Asc(36)
Sql_variant	Not supported

SQL SERVER DF TO SQL MAPPINGS

The following table shows the DF to SQL mappings in CK5 and the allowed new mappings in CK6 for SQL Server 2008/2012.

Current DF – SQL mapping (CK5) SQL Server		New DF – SQL mapping (CK6) SQL Server		
DF Type	SQL Type	DF Type	SQL Type	ODBC Type
DF_ASCII	Char	DF_ASCII	**Char** **Varchar** **NChar** **NVarchar**	SQL_CHAR (1) SQL_VARCHAR (12) SQL_WCHAR (-8) SQL_WVARCHAR (-9)
DF_BCD	SmallInt/Int/Numeric	DF_BCD	SmallInt/Int/Numeric	
DF_DATE	Datetime	DF_DATE	**Date** **Datetime**	SQL_TYPE_DATE (91) SQL_TYPE_TIMESTAMP (93)
DF_DATETIME	Datetime	DF_DATETIME	**Datetime2** **Datetime**	SQL_TYPE_TIMESTAMP2 (-200) SQL_TYPE_TIMESTAMP (93)
DF_TEXT	Text	DF_TEXT	**varchar(max)** **text** **Nvarchar(max)** **Ntext**	SQL_VARCHAR (12) SQL_LONGVARCHAR (-1) SQL_WVARCHAR (-9) SQL_WLONGVARCHAR(-10)
DF_BINARY	Binary/Image	DF_BINARY	**varbinary(max)** **image** **binary**	SQL_VARBINARY (-3) SQL_LONGVARBINARY (-4) SQL_BINARY (-2)

Note: When a type is not supported by the SQL Server version, the CK will automatically change the mapping to a type that is supported by the connected-to SQL Server version.

For example: When DF_Date is configured to be mapped to SQL date. When connecting to SQL Server 2005 (does not have a SQL date type) the mapping will fall back to mapping DF_Date to SQL datetime.

SQL SERVER DATE AND DATETIME CHANGES

SQL Server 2008 introduced new types for date, datetime and time storage. The SQL Server Connectivity Kit version 6 will support these new types.

Several changes have been made in CK6 to better handle date and datetime types in general. Some of these changes may cause backward compatibility issues on existing databases when upgrading from CK5 to CK6.

SQL SERVER DATE AND DATETIME TYPES

SQL Server 2008/2012 has the following date and time types:

- time(0) – time(7)
- datetime
- datetime2(0) – (7)
- smalldatetime
- datetimeoffset(0) – (7)
- date

Microsoft recommends using the **time**, **date**, **datetime2** and **datetimeoffset** data types for new work. **Datetime** and **smalldatetime** are no longer recommended.

TIME

Time columns can have a fractional part with a length of 0 to 7. Time(0) has no fractional part, it is specified as *hh:mm:ss.* Time(7) can have 7 fractional digits (nano seconds) specified as *hh:mm:ss,fffffff*

In CK6 time columns will be handled as follows:

Time columns are shown as DF_ASCII columns.

When an invalid time value is entered that will not be accepted by the backend, an Invalid_Time error will be raised.

When storing or retrieving time columns from a VDF program, the format as specified by the windows regional settings must be used:

For example:

With regional settings set to Dutch(Netherlands) the decimal separator must be a comma.

Move “22:12:13,1234567” To Table.Time7Column

With regional settings set to English(United States) time values will be returned with AM/PM.

Showln Table.Time7Column

Will be shown as

“10:22:13.1234567 PM”

DATETIME

The SQL Server datetime type consists of a date and a time and a fractional part of max 3 (milliseconds). The lowest possible value for a SQL Server datetime column is 1753-01-01.

In SQL server versions before SQL Server 2008, datetime was the only type available for both date and datetime columns. The Connectivity Kit would map both DF_Date and DF_Datetime type columns to SQL datetime.

In CK6 the datetime type can still be used and will function as before with earlier versions of the Connectivity Kit

Datetime values are formatted following windows regional settings. (Date separator, date format, decimal separator, time format (24 hour or 12 hour)

DATETIME2

The datetime2 type has the general format *yyyy-mm-dd hh:mm:ss.fffffff*. The fractional part can be 0 to 7. Datetime(0) has no decimals, Datetime(7) has 7 decimals.

In CK6 datetime2 columns will be handled as follows:

Datetime2 columns are shown as DF_DATETIME columns.

When an invalid datetime value is entered that will not be accepted by the backend, an Invalid_Datetime error will be raised.

When storing or retrieving datetime columns from a VDF program, the format as specified by the windows regional settings must be used:

For example:

With regional settings set to Dutch (Netherlands) the date format must be European, the time format must be 24hour, and the decimal separator must be a comma.

Move “31-12-2012 22:12:13,1234567” To Table.Datetime7Column

With regional settings set to English(United States) the date format must be US, the time format 12hour (AM/PM notation) and the decimal separator must be a point.

Showln Table.DateTime7Column

Will be shown as

“12/31/2012 10:22:13.1234567 PM”

Note:
The VDF runtime datetime type can handle only 3 decimals, where the datetime2 type can store 7 decimals. As a consequence storing or retrieving datetime2 values with more than 3 decimals will be truncated.

DATE

The SQL Server date type can be used to store date values (dd/mm/yyyy).

The SQL Server date type will be mapped to DF_Date

When storing or retrieving date columns from a VDF program, the date format as specified by the windows regional settings must be used.

DATETIMEOFFSET

This is another datetime type that is basically the same as the datetime2 type, but in addition it can specify a timezone offset.

The format is:

YYYY-MM-DD hh:mm:ss[.nnnnnnn] [{+|-}hh:mm]

A time zone offset specifies the zone offset from UTC for a **time** or **datetime** value. The time zone offset can be represented as [+|-] hh:mm:

- hh is two digits that range from 00 to 14 and represent the number of hours in the time zone offset.
- mm is two digits, ranging from 00 to 59, that represent the number of additional minutes in the time zone offset.
- + (plus) or – (minus) is the mandatory sign for a time zone offset. This indicates whether the time zone offset is added or subtracted from the UTC time to obtain the local time. The valid range of time zone offset is from -14:00 to +14:00.

In CK6 the Datetimeoffset type will be mapped to DF_Datetime

SMALLDATETIME

This type stores datetime values in the general format: *yyyy-mm-dd hh:mm*

Note this type does not store seconds.

In CK6 smalldatetime is mapped to the df_datetime types.

The Connectivity Kit will return seconds as 00. Seconds will not be stored on save.

DATE AND DATETIME MAPPINGS

SQL Server versions before SQL Server2008 did not have a separate date type. Date values were stored in a datetime type column, where the time portion would be empty.

SQL Server CK version 5 did not use the SQL date type. DF_Date columns were mapped to SQL datetime columns. With CK version 6 DF_Date columns can be configured to map to either date or datetime.

This mapping change may have consequence for existing databases/applications when upgrading from CK5 to CK6.

The following table shows how DF_Date and DF_Datetime types are mapped to SQL Server types

DF – SQL mapping (CK5) SQL Server		DF - SQL mapping (CK6) SQL Server	
DF Type	SQL Type	DF Type	SQL Type
DF_DATE	**Datetime**	DF_DATE	**Date** **Datetime**
DF_DATETIME	**Datetime**	DF_DATETIME	**Datetime2** **Datetime**

The mappings will be applied only for new columns. Restructuring existing columns will keep their existing types.

In CK6 the mappings can be configured with the MAP_DFDATE_TO_SQLTYPE and MAP_DF_DATETIME_TO_SQLTYPE settings or attributes.

When a type is not supported by the SQL Server version, the CK will automatically change the mapping to a type that is supported by the connected-to SQL Server version.

For example: When DF_Date is configured to be mapped to SQL date. When connecting to SQL Server 2005 (does not have a SQL date type) the mapping will fall back to mapping DF_Date to SQL datetime.

DUMMY ZERO DATE

The CK uses the concept of dummy zero dates to handle empty/null/zero date and datetime values.
A dummy zero date will be used if an empty/null/zero date / datetime is stored in the database and the backend column is defined as not-nullable.
In that case the lowest possible value for the type will be stored in the data base as the dummy zero date.

In earlier CK versions dummy zero date was only used for DF_Date type columns.

In CK6 the dummy zero date will also be used for DF_Datetime columns.

- All DF_Date and DF_Datetime types will use the DummyZeroDate
- DummyZeroDate will be the lowest possible value for the SQL type
  For SQL Server:
  Date = 0001-01-01
  Datetime2 = 0001-01-01
  Datetime = 1753-01-01
  smalldatetime = 1900-01-01
- Move 0 or '' <empty string> To Date/Datetime type will set the Default_dummy_zero_Date.

CONVERTING SQL SERVER DATE AND DATETIME VALUES

The example above converts the native type of DF_Date columns from datetime to date and the native type of DF_Datetime columns from datetime to datetime2.

Since the lowest possible value for the datetime type (1753-01-01) is different from the lowest value for date and datetime2 (0001-01-01), the default value for the columns will also be changed from 1753-01-01 to 0001-01-01.

Important!

The conversion from datetime to date or datetime2 will also convert all dummy zero date values from 1753-01-01 to 0001-01-01.

This is important to realize. The conversion will actually change the data in the database. This may have consequences for existing applications. There are applications that check for the dummy zero date value 1753-01-01 and do something special with it. This can be VDF applications (embedded SQL) or external applications (VRW, Crystal, SQL or any third party application)

SQL SERVER TEXT AND BINARY CHANGES

The following table shows how DF_Text and DF_Binary types are mapped to SQL Server types

DF – SQL mapping (CK5) SQL Server		DF - SQL mapping (CK6) SQL Server	
DF Type	SQL Type	DF Type	SQL Type
DF_TEXT	**Text**	DF_TEXT	**varchar(max)** **text** **Nvarchar(max)** **Ntext**
DF_BINARY	**Binary/Image**	DF_BINARY	**varbinary(max)** **image** **binary**

The text, binary and image type are no longer recommended.

The recommended types to use are varchar(max) and varbinary(max). These types are supported on SQL Server 2005 and later.

APPENDIX B: DB2 TYPE CHANGES

DB2 10.1 TYPES TO DF MAPPINGS

The following tables show how the DB2 10.1 native types map to a DF type.

All types shown are supported by CK6. The CK is able to store and retrieve from/to all the mentioned data types.

DB2 type	DF Type
NUMERIC	Num 0.1 – Num 14.8
DECIMAL	Num 0.1 – Num 14.8
INTEGER	Num 9.0
SMALLINT	Num 5.0
BIGINT	Num 14.0
FLOAT	Num 14.8
REAL	Num 14.8
DOUBLE	Num 14.8
DATE	Date(10.0)
TIME	Asc(11)
TIMESTAMP	Datetime (23.0) – Datetime(23.6)
CHAR(1-255)	Asc(1 – 255)
VARCHAR(1-255)	Asc(1 – 255)
VARCHAR(256-32672)	Text(256-32672)
LONG VARCHAR(1-255)	Asc(1 – 255)
LONG VARCHAR(256-32000)	Text(256-32000)
CLOB	Text(16383)
XML	Text(16383)
CHAR FOR BIT DATA(1-254)	Binary(1-254)
VARCHAR FOR BIT DATA(1-32672)	Binary(1-32672)
LONG VARCHAR FOR BIT DATA	Binary(1-32000)
BLOB	Binary(16383)
GRAPHIC(1-255)	Asc(1-255)
VARGRAPHIC(1-255)	Asc(1-255)
VARGRAPHIC(255-32672)	Text(256-32672)
LONG VARGRAPHIC(256-32000)	Text(256-32000)
DBCLOB	Text(16383)

DB2 DF TO SQL MAPPINGS

The following table shows the DF to SQL mappings in CK5 and the allowed new mappings in CK6 for DB2 10.1.

Current DF – SQL mapping (CK5) DB2	New DF – SQL mapping (CK6) DB2

DF Type	SQL Type	DF Type	SQL Type	ODBC Type
DF_ASCII	Char	DF_ASCII	**Char** **Varchar** **Graphic** **Vargraphic**	SQL_CHAR (1) SQL_VARCHAR (12) SQL_GRAPHIC (-95) SQL_VARGRAPHIC (-96)
DF_BCD	SmallInt/Int/Numeric	DF_BCD	SmallInt/Int/Numeric	
DF_DATE	Date	DF_DATE	Date	SQL_TYPE_DATE (91)
DF_DATETIME	Timestamp	DF_DATETIME	Timestamp	SQL_TYPE_TIMESTAMP (93)
DF_TEXT	Varchar	DF_TEXT	**Longvarchar** **varchar** **clob** **longvargraphic** **vargraphic** **dbclob**	SQL_LONGVARCHAR (-1) SQL_VARCHAR (12) SQL_CLOB (-99) SQL_VARGRAPHIC (-96) SQL_LONGVARGRAPHIC (-97) SQL_DBCLOB (-350)
DF_BINARY	Varchar for bit data	DF_BINARY	**Long varchar for bit data** **Varchar for bit data** **Char for bit data** **Blob**	SQL_LONGVARBINARY (-4) SQL_VARBINARY (-3) SQL_BINARY (-2) SQL_BLOB (-98)

Note: The 'Graphic' types are the DB2 Unicode types.