Team Developer 5.1 Release Notes.

Changes in structure mapping external functions

There is a situation where a string is mapped to a byte[] in an external function. We do provide char[] mapping in an external function and this can be treated as ASCII data.
In Team Developer 5.1, since a string is unicode, we need to convert the actual data to a unicode string. If the external structure is mapped to char[], Team Developer runtime will assume that the data is in asci or local code page and do the appropriate conversion. If it is mapped to byte[], then the data will be considered as binary and no conversion will be done.

Include library and Breakpoint details:

Old serialized data for include library and breakpoint details are discarded while migrating pre-5.1 applications. Pre 5.1 binary applications will loose breakpoint details after migration.

Binary data binding:

Some users may have stored binary and text data in a BLOB datatype in SQLBase. Since there is no way to identify the content of a BLOB column in SQLBase, Team Developer runtime can not convert the text data to unicode data when users bind string variables to BLOB column.
In Team Developer 5.1, we treat the BLOB column as a true binary type and retrieve the data as is (i.e. no conversion at runtime). This could break the old application and this can be avoided by folowing these methods.

Method 1: User is to convert the data to unicode using the SalStrToWideChar api after data is retrieved from the database.
Method 2: Users should change the column type from BLOB to some type which supports unicode. This will trigger Team Developer runtime to do the appropriate conversion.

Migration:

When migrating applications to Team Developer 5.1 it should be standard operating procedure to first perform a backup of your application and associated files before migrating. Also ensure that the files involved have write permissions before migrating.

Opening a 5.1 application in a previous version:

If you want to open a Team Developer 5.1 application in a previous version of Team Developer the following steps need to be followed:

- Open your Team Developer 5.1 application and save as an apt.
- Open this apt in notepad.
- Change the outline version (i.e. 4.0.37) to the corresponding outline version for the Team Developer version you want.

The table below lists the outline versions which corresponds with various Team Developer versions.

Team Developer Version	Outline Version
1.1.x	4.0.26
1.5.x	4.0.27
2.0.x	4.0.28
2.1.x	4.0.28
3.0.x	4.0.31
3.1.x	4.0.32
4.0.x	4.0.34
4.1.x	4.0.35
4.2.x	4.0.35

- Choose 'Save as...' option in Notepad.
- Select "ANSI" in the encoding combobox.
- Save the file.

SalStrBufferLength

nLength = SalStrGetBufferLength(strString)
nLength is the length of the strString buffer in bytes.

Team Developer uses 16bit characters from version 5.1 onwards, all the Team Developer applications developed with prior versions of Team Developer using this API will show a change (i.e. increase) in the function's return value.

SalTblPopulate

SalTblPopulate does not support the display of long column data for OLE DB connectivity.

Note: This is a limitation from previous versions of Team Developer.

SqlSetLongBindDatatype

It is now mandatory to call "SqlSetLongBindDatatype(bindvarindex,bindtype)" in applications where there is code that binds blob data using long string variables. This needs to be done on both insert and fetch operations.

Inserting Unicode data in SQL Server

When inserting unicode data to a Microsoft SQL Server database you need to prefix an "N" to denote that you are passing unicode data. Otherwise that data would be treated as ASCII by your database. Example: insert into tableX(fname,lname) values (N"Peter",N"Ryan")

Passing Unicode data with Oracle

With the Oracle database you need to set the NLS_LANG environment variable on the client side in order to pass unicode data.

Note: With Team Developer 5.1 SP2 and above this is no longer a requirement.

Binding

It is important to use the right data type when binding columns containing string data. For example you need to use a string variable when binding to a char/varchar/nchar/nvarchar column and use long string variable when binding to long column types such as long/long varchar/long nvarchar etc.

Compiler Support for Extended Symbols

If you want the compiler to support the following extended symbols:

- 2460 ... 24FF (i.e. Enclosed Alphanumerics)
- 3000 ... 303F (i.e. CJK Symbols and Punctuation)
- 3200 ... 32FF (i.e. Enclosed CJK letters and months)
- FF00 ... FFEF (i.e. Halfwidth and Fullwidth Forms)

You will need to add a new DWORD value called **ExtCompilerChars** with a non zero value to the windows registry key HKEY_CURRENT_USER\Software\Gupta\SQLWindows 5.1\Unicode

Note: You are encouraged not to use these types of charcters in your applications as it will slow down compile time. When we talk about use we mean anything which the compiler will compile such as variables, constants, function names etc.