Team Developer 3.1

New features in SQLWindows client application GUI

Table windows extended GUI

A large number of new SAL functions permit you to set and get the following attributes at runtime:

- Font styles for an entire table window, a row, a row header, a column, a column header, or a cell.
- Foreground and background colors for an entire table window, a row, a row header, a column, a column header, or a cell.
- Separator line styles for a row or a column.
- Bitmap images for a cell, row header, or column header.

All the function names begin with **VisTbl** and are documented in SQLWindows online help under topic "Table window extended GUI functions".

Resizeable dialog boxes and toolbars

Dialog boxes have three new attributes:

- Resizeable
- Vertical Scroll - only enabled when the dialog is dockable or resizeable
- Horizontal Scroll - only enabled when the dialog is dockable or resizeable.

Toolbars have one new attribute: Resizeable. It is only enabled when the toolbar is dockable.

These objects, when flagged as Resizeable, present a standard "gripper" cursor to allow the user to drag a border of the object to resize it.

These objects are stateful - if a dialog is free-floating, and you resize it, then dock it, then undock it, it will "remember" the custom size that you gave it when it was last free-floating, rather than retaining its docked size or its design-time size.

New messages give you some control over sizing events:

SAM_DockResize is sent just before a resizing or a change in docking state. You can influence this event by returning a numeric value to this message. The value would contain your desired size for the object. However, in docking situations, there may be competing requests for space from other objects that are located inside the same dock bar as the object you wish to resize. For example, it’s common for two or more toolbars to share the same dock bar. There may also be other dock bars active in the parent window, with other objects inside them. There is a process of negotiation that occurs when there are competing requests, and negotiation does not guarantee that the size you requested will be granted.

It is unsafe to call certain functions involving window size or location from within the context of SAM_DockResize. Read SQLWindows online help or *Developing with SQLWindows* for more details.

SAM_DockResizeNotify is sent after a sizing or docking state event has completed. This is your opportunity to rearrange the child windows within your object to accomodate the new size.

New features in SQLWindows developer IDE

Debugging and breakpoint enhancements

A new breakpoint management dialog box shows all breakpoints that are currently flagged in the application. A checkbox permits you to enable/disable the breakpoints from within the dialog.

In addition to breakpoints on a line of code, as in previous versions of SQLWindows, the breakpoint management dialog also permits you to enter data expressions. When the value of such an expression changes, execution pauses and a message box informs you of the change.

The breakpoint management dialog contains a "number of iterations" field that you can use to indicate that you want the breakpoint to be suppressed until that number of iterations has occurred, then break.

For a breakpoint anchored to a line of code, you can also specify a condition (a data expression) to determine whether or not the breakpoint will actually break.

The Preferences dialog previously allowed you to set a different outline text color for lines with breakpoints. Now there are two colors available, one for enabled breakpoints and one for disabled breakpoints.

When a breakpoint occurs, you can use the Step Over and Step Into operations that were available in previous versions of Team Developer. You can also use two new operations. Step Out executes until control passes to the outline level higher than the current breakpoint. Run to Cursor executes until it reaches the line of code that you have highlighted, then breaks again.

More SQLWindows new features

Event logging

Event logging allows an application to automatically log an event and optionally continue running, rather than displaying a runtime message box that needs to be answered through human intervention. Events such as SQL errors, array index errors, etc., can now be logged. This is particularly advantageous for applications that run unattended, such as COM servers.

To activate logging, call function **SalUseEventLog**. See SQLWindows online help or the Function Reference manual for detailed syntax information. Output goes to the Windows event log, or (for Window 98 and ME, which don’t support event logs) to a file designated in a registry entry.

The "continue" option in SalUseEventLog behaves as follows:

Message box button choices	**Behaves as if**
Yes / No	You clicked Yes
Abort / Retry / Ignore	You clicked Abort
OK / Cancel	You clicked OK

Tracing

With tracing, you can direct detailed diagnostic information to several different output locations:

- The Windows event log (for Windows 98 and ME, a data file is used since the event log isn’t supported by these versions.)

- A named file.
- The SQLWindows output window normally used for displaying compiler errors. Note that this option is only available when the application is running in Debug mode.
- Directly to the "stdout" window, so that trace output can be integrated with third -party debugging tools.

For more information, see online help and books for functions SalStartTrace, SalEndTrace, and SalTrace.

XML support for table windows

New functions in SQLWindows permit you to:

- Write out the full or filtered contents of a table to an XML document and/or schema.
- Read back an XML document and schema into a table window.

New functions include

Function	Description
SalTblWriteXMLandSchema	Write table window information (all rows) to an XML document and/or and XML schema.
SalTblWriteXMLandSchemaEx	Write table window information (selected rows) to an XML document and/or and XML schema.
SalTblSetFromXMLSchema	Validates that an XML schema matches a table window column layout
SalTblPopulateFromXML	Reads data from an XML document into the cells of a table window

New messages include:

Message	Description
SAM_WriteXMLRow	Sent just before a row is written to an XML file.

See SQLWindows online help under the index entry "XML" for a list of functions, constants, and messages. Also see the XML Support section in Chapter 15, *Table Windows*, in the book *Developing with SQLWindows*.

COM+ performance improvements

New function **SalComCleanupMode** allows you to choose when to release resources allocated to COM server objects:

As soon as the last object on a specific thread is destroyed (early cleanup, current behavior).

When the thread itself is destroyed (late cleanup, new alternative).

By waiting until the thread itself is destroyed, you can avoid the time expense of initializing and destroying the resources repeatedly. The time savings can be very significant in a COM server application that is called repeatedly by a client.

The default is early cleanup, since that is the method used in earlier versions of SQLWindows.

There have also been several internal improvements in object creation, function invocation, and object initialization and allocation. Cumulatively these improvements add up to noticeably quicker performance.

Enhanced OLE DB features

SqlUDL is a system variable that can contain the name of a UDL file to use for OLE DB connection information. This variable was introduced in version 3.1. One of its purposes is to ease the migration of existing SQLWindows applications from use of native routers to use of OLE DB.

To make this easy, function **SQLConnect** has been altered in SQLWindows version 3.1. SQLConnect now looks first at variable SqlUDL and, if it finds a file name in that variable, reads connection information from that file. If it finds a provider name in SqlUDL, it uses the provider name. If the database name or user name or password was not specified, SQLConnect will obtain the needed value from the values of variables SqlDatabase, SqlUser or SqlPassword. It forms a connection string, then makes an OLE DB connection with that string. If SqlUDL is null, SqlConnect uses the older (API and routers) method of connecting with the values of SqlDatabase, SqlUser, and SqlPassword. So, in many cases, existing apps simply need a few lines to set the value of SqlUDP and the rest of the app will run smoothly against OLE DB.

New message **SAM_SessionError** has been added to make it easy to know when a SQL error is originating from an ordinary connection or from a session. Ordinary connections will continue to use SAM_SqlError.

New function **SqlGetLastStatement** shows the text of the last statement executed. It is global, not dependent on cursor, so be aware of the timing of commands when using this function.

Miscellaneous enhancements

Previously undocumented SQLWindows functions SalGetWindowLabel, SalPause, and SqlGetCursor have now been documented.

The menu item Edit, **Replace** now has an accelerator key, Ctrl+R.

Report Builder enhancements

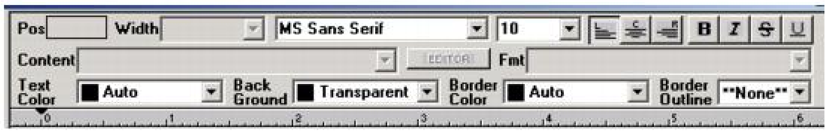
Use of multiple paper trays

On the dialog invoked by menu item Report, Format, Report, there is a new tab named Paper Source. On this tab you can choose a paper tray for the first page of the report and a different tray for all other pages. These preferences automatically adjust for the printer selected. If you change printers and the selected tray is not available on the new printer, the default tray will be used instead.

You get a second opportunity to choose trays when you invoke the Print dialog. Changes made here will not persist, but changes made in the Report/Format/Report dialog do persist.

New formatting controls in the ribbon toolbar

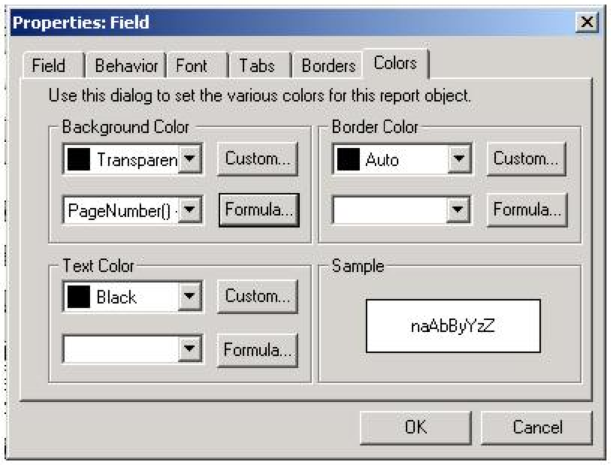
New controls at the bottom edge of the ribbon toolbar allow you to choose text color, background color, border color and border outline for report objects without invoking their property dialogs.



New abilities to use conditional formatting

Some elements of a report have been enhanced to include a control for selecting a formula (conditional expression). Properties that are subject to conditional formatting include:

- Text color (default: black)
- Background color (default: transparent)
- Border color (default: black)
- Font (default: Times New Roman)
- Font style (default: Regular)



You are not required to enter a formula at all when setting these properties. If you do not enter one, your chosen property value will always be used. But if you do enter a formula, the formula is evaluated at runtime. If it is TRUE (non-zero for numeric formulas, non-null for strings) then your selected property value is used. If it is FALSE, the default value of the property is used.

All five of the properties above can be conditionally formatted for fields. For lines, only background color and border color can be conditionally formatted.

Manual or conditional page break

In the Behavior tab of the Line properties dialog, there is a new option to choose an expression or formula to be associated with a page break. If the formula evaluates to TRUE, the page break will occur; if it is FALSE, the page break will not occur. It is still possible to specify a page break that is unconditional (always breaks).

Calculate total pages

New function **TotalPages()** can be invoked to determine the total number of pages in the report, even at the beginning of printing. This is useful for creating "Page 1 of 15" style strings for headers and footers.

International language date display

A new option for menu item Report, Format, Report allows you to click a checkbox titled "Enable Intl. Date Display". If you do so, function CurrentDate() will return a date string formatted in the workstation’s locale language. In addition, new function **DateToStrPictureIntl()** will retun a string formatted in the workstation’s locale language, regardless of the setting of the new checkbox.

Enhanced Oracle router

The router has been upgraded from use of Oracle OCI 7 APIs to optional use of OCI 8 APIs. This permits you to use some of the Large Object (LOB) datatypes available in OCI 8. Previous versions of the router used the Long Raw datatype, but since Oracle has deprecated this datatype beginning with OCI version 10, it is important that you have enough time to move your datatypes to LOB while still being able to run applications that use Gupta routers.

The router determines which APIs to use based on a new SQL.INI keyword. **USELOB** has a default value of 0 (older APIs will be used). You can also set its value to 1 (OCI 8 APIs will be used). LOB datatypes work only when USELOB=1.

Another way of getting and setting this option is through the use of new parameter **DBP_ORAUSELOB** with the SqlGetParameter and SqlSetParameter funcitons.

LOB datatypes supported include CLOB and BLOB.

Web Application Manager (WAM) for LINUX

Some of the WAM components have been adapted for use on Linux servers running the Apache web server 1.3.x or 2.x. The CGI and DSO components now reside there, along with a new component, the Gupta Naming Services daemon. This means that Team Developer applications can now use Apache web servers as well as Microsoft web servers. For detailed information on installation and configuration of these components, see the chapter *WAM for Linux* in *Building Web Applications with Gupta*.

SQLBase version 8.5 integration

Team Developer 3.1 ships with SQLBase version 8.5. There are many major new features in 8.5, all of which are described in the book *SQLBase Guide to New Feaures*. In this section we will only discuss the features that are of greatest interest to client application developers using Gupta SQLBase or Gupta routers.

Multiple SQLBase Installations

SQLBase now has the ability to support more than one installation of SQLBase on a computer. Multiple instances of the SQLBase database engine, even different versions of SQLBase, can run simultaneously. Multiple client configurations can also run simultaneously without interfering with each other.

The configuration differences that make multiple installation possible are only available in SQLBase 8.5. You can run one or more SQLBase 8.5 database engines concurrently with one earlier version of SQLBase, but you cannot run mutliple earlier versions simultaneously.

SQLBase configuration file (SQL.INI)

In order to support multiple installations, the configuration file, always named SQL.INI in versions prior to 8.5, now has a flexible name and path specification. You may use whatever name you like in place of SQL.INI. *We will continue to use the name SQL.INI throughout SQLBase documentation*, although your actual file name may be different.

Client applications written with Team Developer also use the configuration file to determine what database servers and communication protocols are available. With the possibility of multiple configuration files and multiple servers active on a single machine, there is a need for the client application to indicate which configuration it wants to use. This need exists both at design time and at run time.

The selection of a configuration file at design time is handled by a new option in the Preferences dialog of SQLWindows. The General tab of that dialog contains a control that allows you to type or browse a specific filename. This choice is used by SQLWindows and by other Team Developer design tools, such as SQLTalk, Report Builder, and Team Object Manager.

These tools can also accept a command line argument specifying what configuration file to use. See the book for each tool for exact syntax descriptions for the argument.

SQLWindows also provides a means of specifying a configuration file at runtime. An application can supply a simple or fully qualified filename to new system variable **SqlINI**, and that configuration file will be used when making database connections.

A simple filename, without path, causes SQLWindows to search for that filename in the current path of the application. A fully qualified file name causes SQLWindows to search only in that specified directory.

If you do not make an explicit configuration file choice via the Preferences dialog or the SqlINI variable, Team Developer will use the method of locating SQL.INI that was used in previous versions.

Variable SqlINI, once changed, affects all future connections, through both SqlConnect and SqlCreateSession. If a connection is already open when SqlINI is changed, the next call that references that connection will return an error. It is strongly recommended that open connections be closed before changing the value of SqlINI.

The value of the current configuration file can be retrieved through parameter DBP_SQLCONFIGFILENAME in conjunction with function SqlGetParameter. This works whether SqlINI has a non-null value or not.