Release Notes **Gupta Team Developer 3.0**

New features in Gupta Team Developer 3.0
- New debugging tools
- Active Coding Assistant
- Interface Inheritance
- Docking windows and toolbars
- Flat push/option buttons
- Report Builder command-line arguments
- Router enhancements
- New version of image management
- Design Hook Interface

Notes for Beta Testers

Migration Notes
- Outline Version Change
- COM Servers
- Window size change
- Window Class Name change
- Registry change

Known Problems, Limitations and Workarounds
- Installation
- Miscellaneous
- Web Applications
- COM Events
- COM Servers
- Connectivity Limitations

New features in Gupta Team Developer 3.0

Team Developer 3.0 is designed to improve your productivity and product quality when creating applications, especially applications that will run in a COM+ and/or .NET environment.

The new Interface Inheritance feature allows more comprehensive use of COM clients and servers.

COM+ performance has been improved. A new ability to invoke COM server methods by dispatch ID instead of name makes COM clients faster.

The new Active Coding Assistant provides instant, detailed help as you edit your application, displays lists of possible values in the context of your current statement, and optionally auto-completes words and phrases for you.

The new Active Debugging Assistant greatly increases the types and amount of debugging information available to developers during a debugging session, and makes it easier to monitor and modify variable values.

The user interface now features "flat" push/option buttons.

You can now design toolbars and dialog boxes that dock and undock to parent windows, either through user manipulation or programmatically.

The design environment now allows mouse wheel scrolling in both of the outline panes.

New debugging tools

There are several new ways to view and edit variables at runtime. These new tools make debugging much faster and simpler.

Team Developer 2.1 allowed you to view and change the value of a variable at runtime, or specify new variables, using the Expressions window and Variables window. You can still do these tasks in the same way in 3.0, but you can also type directly into the cells in the Variable window. Type a new variable name in a blank cell in the "Name" column to add a variable. Type over the value of an existing variable to change its value at runtime.

There are also two new variable watch windows: Auto and Locals. Auto shows those variables that are present in the current line of code (the breakpoint line) and the line of code preceding the current line. Locals shows only those variables that are in "local scope" at the time of the breakpoint. These new windows are accessed from the Debug toolbar.

All the watch windows now handle complex user-defined-variables (UDVs) more smoothly. A "+" icon allows you to expand the UDV variable name in a tree view to see nested variables within the UDV.

Active Coding Assistant

New in 3.0, the Active Coding Assistant is a background process that can increase your coding productivity up to 20% and drastically reduce spelling errors. It provides four main features:
Complete Word - Types the rest of a variable, command, or function name once you have entered enough characters to make the expression unambiguous.
Parameter Info - The Parameter Info option opens the Parameters list to give you information about the number, names, and types of parameters required by a function or attribute, plus the parameter description if available.
Quick Info - The Quick Info option displays the complete declaration for any identifier in your code. Move the mouse so that the cursor is over an identifier (hover over an identifier), and you will see its declaration displayed in a yellow pop-up box.
List Members - Displays a list of valid member variables, functions or symbols for the appropriate class or scope. Selecting from the list inserts the text into your code.

You can configure the Active Coding Assistant from the Preferences dialog to control the behavior of individual features.

Interface Inheritance

Many widely available COM servers use interface inheritance. For example, the Microsoft XML parser, MSXML, has a large number of interfaces that inherit from the IXMLDOMNode interface. To get the most effective use out of COM servers like these, the SQLWindows ActiveX Explorer must create APL files with class hierarchies that reflect the inheritance hierarchy of the server. In Team Developer 3.0, ActiveX Explorer has been enhanced to recognize and preserve interface inheritance in COM servers. For this reason, as well as others, Gupta recommends that you regenerate any existing ActiveX APLs using version 3.0.

A related enhancement allows you to specify interface inheritance in the COM servers that you create using SQLWindows. The COM Class Wizard, context menus, and Coding Assistant have been modified to present a Derived From property for interfaces that you are editing.

If you are using interface inheritance in a COM server that will be accessed by Visual Basic version 6 or Visual Basic .NET, make sure that the VB applications use late binding when declaring objects from that COM server. Otherwise VB will produce errors at runtime. Force late binding in VB by doing two things:

- Declare the variable that will contain an object from the COM server as type Object, not the specific type from the server.
- Declare the variable first, as a new Object; then assign the COM server reference to it later in your code.

```
Dim objIslandSales as New Object
Set objIslandSales = CreateObject("IslandSALESInfoSVR.ISOSalesServer")
```

Docking windows and toolbars

The Attribute Inspector for all top-level windows now features two new attributes: Allow Child Docking and Docking Orientation. With these two attributes you can specify whether you want to allow dialog boxes to dock to the top-level window, and which side(s) of the window should be used when docking. Similar attributes for the dialog box (Allow Dock to Parent, Docking Orientation) let you specify whether it is capable of docking, and on which sides. You control this behavior with new messages SAM_Dock and SAM_DockChange, and new functions SalDlgGetDockStatus, SalDlgSetDockStatus, and SalWindowGetDockSettings.

Toolbars are also capable of docking and undocking within their parent windows, and the Attribute Inspector for toolbars contains two new attributes (Docking Toolbar and Toolbar Docking Orientation) to control this behavior.

Flat push/option buttons

A new attribute for pushbuttons allows you to set their appearance to Standard or Flat. Flat buttons are drawn without borders, resembling the toolbar buttons used in Microsoft Office applications.

Report Builder command-line arguments

New arguments for REPBI30.EXE allow you to supply a database name, user ID, and password. With these arguments, reports that require database access can now run without user intervention. In addition, two more new arguments allow you to specify output to the default printer, or to a printer of your choice. See the online book *Business Reporting*, chapter 1, for details.

Router enhancements

In version 3.0 the Gupta router for Oracle allows you to indicate that you wish to connect to a database in the "sysdba" role. You do this from Connectivity Administrator by altering the Oracle connection string slightly, as shown in this example line from SQL.INI:

remotedbname=ora9i,@ora9i as sysdba

Thereafter, you can set the user ID as "sys" and the router will actually connect with "sys as sysdba".

If you are connecting to Oracle using an OLE DB data source, these steps are not required; "sys" will connect with sysdba privileges.

New version of image management

Team Developer 3.0 is built using ImageMan version 7.2. The new DLLs which support this version are imhost32.dll and imgman32.dll. Deployment files affected by this change are im31bmp.dil, im31gif.dil, im31img.dil, im31ipj.dil, im31pcx.dil, im31tif.dil, and im31wmf.dil.

Design Hook Interface

In previous versions of SQLWindows, you could set design time hooks in SQLWindows to enable the SQLWindows design environment to notify your tools about certain events. The hooks were set by specifying a DLL name in a registry entry of the form:

```
DesignHookDLLn = MYAPP.DLL
```

…where n is a sequential number. MYAPP.DLL would be loaded when SQLWindows was launched.

In version 3.0, this capability still exists, but the registry entry format has changed. Rather than using sequential numbers, the registry entries now are identified by unique string values. The registry entries are stored under this key:

"HKEY_CURRENT_USER\Software\Gupta\SQLWindows 3.0\DesignHookDLL".

Because of the format change, the function CDKGetDesignHookOrdinal has been rendered obsolete. The function CDKSetDesignHook now takes a string parameter, and CDKClearDesignHook also now takes a string parameter. See the online book *Extending the Gupta Development Environment* for detailed information about these functions.

Notes for Beta Testers

Even though you may have used 3.0 in beta, we still recommend that you discard the 3.0 applications you have been working with and re-migrate your applications using text APT files, because internal outline modifications may have continued throughout the beta.

Migration Notes

Outline Version Change

The features in 3.0 have resulted in a new outline format. Source files opened in SQLWindows (and all included libraries) will be converted to the new format. Converted source will not be useable in a prior version of SQLWindows. Therefore, Gupta **STRONGLY RECOMMENDS THAT YOU BACK UP YOUR EXISTING SOURCE FILES PRIOR TO USING SQLWindows 3.0.**

COM Servers

Gupta **STRONGLY RECOMMENDS THAT YOU UNREGISTER SQLWindows COM SERVERS PRIOR TO INSTALLING SQLWindows 3.0. Not doing so may result in COM Servers not being able to be unregistered or registered properly.**

It is impossible for a SQLWindows 3.0 COM Client application to utilize a SQLWindows 2.x "inprocess" (DLL) COM Server and it will cause unpredictable results. This restriction does not apply to Utilizing SQLWindows 2.x "out of process" (exe) COM Servers, although it is not recommended.

Window size change

Prior to version 3.0 of SQLWindows the attributes Height and Width, for top-level windows, referred to the overall window size, including the client area and the non-client area (title bar, status bar, and borders). In Windows XP, the non-client area elements have a larger default size than in other versions of Windows. Since the overall window size stays the same, this results in less client area, and potential

"clipping" of child window elements in your application. So if you developed an application in SQLWindows 2.x or earlier, under Windows 2000, for example, then upgrade to Windows XP but still retain SQLWindows 2.x, you will see this smaller client area. This might introduce scroll bars in client areas that previously didn't have scroll bars. This effect is related to the operating system, not the version of SQLWindows.

In 3.0, to improve the consistency of SQLWindows applications across versions of Windows, the Height and Width attributes were changed to refer to the client area, so that the client area of a top-level window could remain constant under any Windows version. Functions SalGetWindowSize and SalSetWindowSize are unaffected by this change.

Using the height value in SQLWindows 2.x and earlier (which represented the overall window size), and applying it to the client area in SQLWindows 3.0, would result in a significantly larger client area and a larger overall window, one that might no longer fit the desktop. *For this reason, an outline that is migrated to 3.0 undergoes a reduction in the height attribute of every top-level window*. SQLWindows 3.0 queries the operating system to get the height of the non-client elements, then reduces the old top-level window height attribute by that amount. The result is a top-level window with exactly the same client area height that it had in the earlier version of SQLWindows.

It is essential to perform this outline migration using the same Windows environment that was used to design the application in version 2.x or earlier. As noted, non-client elements have a different default size in XP than in earlier versions, and it is also possible to set your particular Windows environment to some other, non-default value for sizes of title bars, status bars, etc. When the migration facility of SQLWindows 3.0 queries the operating system to get the heights of the non-client elements, those heights must be consistent with the ones used when the original window height was set. So you must not take an application designed using SQLWindows 2.1 and Windows 2000, and migrate it directly to SQLWindows 3.0 running under XP. Do the migration to SQLWindows 3.0 using the original operating system, then move to the new operating system after the migration is complete.

Remember, the overall goal of the migration facility is to keep the client area of top-level windows consistent with their size in earlier versions of SQLWindows. If your window filled the desktop under Windows 2000, then keeping the client area the same in Windows XP means that your overall window will now be slightly too big for the desktop. So even after a correct migration you may still need to modify the layout of some of your windows.

SalGetWindowSize and SalSetWindowSize deal with the entire window area, including the non-client area. So if you use these functions on a given window under different operating systems, you can expect to see different results, even though the design-time window size specifications were the same in all cases.

Window Class Name change

Prior to version 3.0, SQLWindows window objects have a Windows class name with a prefix of "Centura:". (ex. Centura: Form, Centura: Toolbar, etc.) This prefix has been changed to "Gupta:". Applications that utilize WINAPI functions and require a window class name, such as *FindWindowA*, will need to be changed appropriately.

Registry change

Prior to version 3.0, Registry information was stored in \\HKEY_CURRENT_USER\SOFTWARE\CENTURA\.
This has been changed to \\HKEY_CURRENT_USER\SOFTWARE\GUPTA\.

Known Problems, Limitations and Workarounds

Installation

- If an error occurs during the installation with the text "You do not have access to make the required system configuration modifications. Please rerun this installation from an administrator's account", this means that there was a previous version of Gupta Web Application Manager installed. To rectify, uninstall the product. Using REGEDIT.EXE, browse to \\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services. Delete the Key "CenWebAppMgr." Reboot and reinstall.

- Before uninstallation, if SQLBase is running as a service you should manually stop the service.

- After an uninstallation, before doing a reinstall, you should reboot your computer to be sure that no Team Developer DLLs remain in use.

Miscellaneous

- Since the version number of this release has changed, the names of most Team Developer EXE and DLL files have changed in order to avoid potential version conflicts. For example, CDLLI20.DLL or CDLLI21.DLL is now CDLLI30.DLL. If you happen to have linked directly to one of these DLLs in your application (rather than using an APL provided by Gupta), you should change the reference in your application in order to avoid strange behavior and crashes.

- Calls to Team Developer DLLs from non-Gupta applications may fail when used against CDLLIxx.DLL, where xx is 20, 21, or 30. This applies only to applications such as stand-alone C programs that are not executing within the Team Developer runtime environment. By contrast, if your SQLWindows application uses an external DLL, and that DLL in turn calls into CDLLIxx.DLL, this problem will not occur.

- UDVs passed as parameters to top-level Windows cannot be modified by assignment, either to other UDVs of an appropriate type or to the result of calling 'new'. External UDVs known only to an external outline are not fully supported. There are two known issues: SalObjIsDerived() does not work with these type of external objects (objects created in a different outline without any definition in the outline using it), and parameter type checking does not work for functions needing external objects.

- Assignment of Window classes derived from a base Functional class is not supported. This will not generate a compiler error, but will produce indeterminate results.

- MTS component activation within a SQLWindows client EXE is not supported. SQLWindows Client EXEs can only use MTS-registered components that are configured to run within a dedicated server process (Activation: Server Package).

- The Customizer is being deprecated in favor of the Attribute Inspector for modifying object properties.

- Certain samples require the presence of particular ActiveX controls (in particular, the Calendar Control 8.0). The COM servers used by the sample applications are fairly common, but may not be installed on your machine. If attempts to run the samples produce ActiveX error dialogs, you may not have the required server installed, or you may have an incompatible version.

- If a tool bar has been designed as docking-enabled, and the orientation specified for the toolbar is not compatible with the orientation of the top-level window, the toolbar will be created as non-dockable.

- When passing parameters to windows that are created at runtime, as in the SalModalDialog function, you can use entire arrays as receive parameters for the windows. However, individual array elements as receive parameters will cause a compile error.

- When processing the actions of the SAM_DockChange message in a parent window, or the SAM_Dock message in a dockable child window, do not call SalModalDialog, SalHideWindow, SalShowWindow, SalSetWindowLoc, or SalSetWindowSize. Calls to these functions will cause an event loop that will freeze your application.

- The reserved word "this", used by class definitions to refer to a current instance of that class, should not be used as part of a bind variable name or into variable name in a SQL query. It will result in an error at runtime.

Web Applications

- When deploying Web applications, please remember to run the Team Developer runtime installer (deploy30.exe) when you install your application. The Web Deploy installer does not include Team Developer runtime files.

- Netscape Web Servers later than Enterprise version 3.6 cannot be detected by the TD 3.0 installer. Please choose 'Other' as the Web server, rather than 'Netscape', if you are running a server later than 3.6 and 'Netscape Web Server' is not marked by default in the "Web Server Document Directory" installation dialog.

- When first running the Tutorial application in Web mode, you may see an error ("Error in comIslandPRODUCT.Create() : COM Server Failed") due to registration ordering issues if Web App Manager is automatically started as a Service. Restarting the application using AppConsole will resolve this issue.

- In AppConsole, it is not necessary to set Minimum Processes higher than 1 for dedicated applications, since each user will create and destroy processes when they use the application. For reusable applications, Minimum Processes can be set higher, so that performance is increased because these applications do not destroy processes. An optimum number for small to mid-sized applications may be in the range of 10-20. Having large numbers of processes running can result in errors due to running out of resources.

- Starting and stopping AppManager from AppConsole in rapid succession may result in initialization errors that can crash AppManager.

- If a machine running AppManager is shutdown, and the AppManager is being remotely administered from AppConsole, the remote machine may hang until the machine is re-started, or may display an error message like "Information for this Application Service is not available". Acknowledging this error message may cause the application to terminate.

- Microsoft Personal Web Server and IIS 4 have been observed to truncate long POST messages to browsers - this can affect the operation of state variables.

- The Object Compiler cannot be used to compile Web applications on Windows 2000.

- When starting Web App Manager with multiple active applications (i.e. MinProcs > 1) that connect to SQLBase using anonymous pipes (SQLAPIPE), if SQLBase is not already started you may see an error to the effect that a server initialization failed because SQLBase was already running. This is a benign error and can be ignored. It is the result of a minor race condition during the auto-start behaviour of SQLAPIPE.

- In the case of a crash of a web application, system resources can be left open. Use Task Manager to end any orphan processes before restarting Web App Manager.

COM Events

- Events defined in a type library may not show up in a .APL file generated from that type library. In this case, in a client application, they will also not be listed under "Message Actions" for a

COM proxy class derived from the COM proxy class that contains the events. If this occurs, open ActiveX Explorer and click on the "Gupta APL's" radio button. Select the .APL file of interest. If a message appears stating that the registration for the type library is damaged, unregister and re-register the type library from which the .APL was generated, and regenerate the corresponding apl.

- COM Events are implemented as "callback" IDispatch methods where the server calls the client. This IDispatch identifier is known through a UUID. In SQLWindows Message Actions for a COM Proxy, the "friendly" name for the event is displayed to the developer but the UUID is stored as a hidden property. This hidden property is only created during the definition of an event and is never refreshed. Therefore, if changes occur that invalidate this hidden property (such as changing the GUIDs of the COM server or replacing the COM server with another that has the same name but different UUIDs) the message action needs to be removed and added again.

- Events with parameters of either IDispatch or Object are not invoked.

COM Servers

- SQLWindows COM server methods with a parameter datatype of DateTime do not accept values that are prior to January 01, 0100. An OLE Automation message "Error code at invocation of <method name> 80020005" occurs.

- A SQLWindows COM client application passing a Variant parameter containing a SafeArray to a SQLWindows COM server may yield unspecified results.

- A SQLWindows COM client application passing a Variant parameter containing a IDispatch (Object) to a SQLWindows COM server may yield unspecified results.

- If the Word 9.0 APL is generated using ActiveX Explorer and then the Word 9.0 ActiveX object is added to a form window, a GPF will occur on exit from SQLWindows. Two possible workarounds are as follows: 1. Generate the .APL, exit SQLWindows, then re-enter it to edit the application that contains the MS Word object in a form window. The GPF will not happen if the .APL was previously generated. 2. The default .APL generated by adding the Word 9.0 ActiveX object does not cause this problem. Do not use ActiveX Explorer to generate the .APL if it is not necessary to do so.

- A COM server function that returns a Variant that has been set to a blob value gets an OLE Automation error if a SetBlob is not performed within the function and the COM server is "out of process" (EXE.) This can be worked around by: Making the COM server "in process" (.DLL), or calling the GetBlob and SetBlob methods if a parameter of type Variant was passed into the function.

- A GPF may occur when you exit SQLWindows under the following circumstances: 1. You are editing a COM server application that references a missing APL. 2. You regenerate that APL using ActiveX Explorer while editing the app. 3. You make a change that references a class already existing in the app and contained in the APL that has just been regenerated. You build an executable and save the app. To work around this problem, open a new SQLWindows application, regenerate the APL from within that empty new application, and close that application without saving it.

Connectivity Limitations

- You cannot connect to SQLBase from a web application using the anonymous pipes protocol (sqlapipe) unless SQLBase is running as a service. A workaround is the use of the TCP/IP protocol instead.

- Microsoft SQL Server OLE DB Provider and ODBC driver have limitations in converting a BSTR type into a datetime database type. So, if a SQLWindows application is trying to insert a String variable into a datetime column, the provider requires that the datetime format in the String be <yyyy-mm-dd hh:mm:ss>. If the datetime format is different, insert call fails. The problem is that the SQLWindows string datetime format is different. SQLWindows format is <yyyy-mm-dd-hh:mm:ss.msmsms>. Note here the difference is in the hyphen (-) character between the date part and the hour part.

- Array type parameters in stored procedures are not supported with any provider.

- Stored procedure return status currently does not work with Microsoft SQL Server.

- With Microsoft Oracle OLE DB Provider, If the user has only Oracle 7.x client software installed, they cannot use the OLE DB provider to execute PL/SQL Statements that have out parameters. The out parameters are not updated.

- Connecting to Oracle 8.x databases in an MTS environment is quite involved. Please refer to the following Microsoft Knowledgebase articles for instructions on this subject: http://support.microsoft.com/support/kb/articles/Q193/8/93.asp and http://support.microsoft.com/support/complus/mtsandoracle.asp . The Oracle 8.1.7 client works best with MTS.
- The Native router for Ingres is not threadsafe. You should not use Ingres connectivity with 'Single Threaded Apartment' COM servers. Usage in Web applications may also give unpredictable results.

- For OLE DB connections, setting Autocommit ON now forces a rollback of any existing uncommitted work. To avoid unexpected data loss, your applications must be certain to commit before setting on Autocommit.

Copyright © Gupta Technologies LLC. Gupta, the Gupta logo, Centura, and all Gupta products are licensed or registered trademarks of Gupta Technologies, LLC., All other products are trademarks or registered trademarks of their respective owners. Copyright © 2001 and 2002 Gupta Technologies LLC. All rights reserved.