INGRES

Ingres Best Practices: **Upgrading from OpenROAD 3.5 to OpenROAD 2006**

This document will guide you through the steps Ingres Corporation recommends to upgrade an OpenROAD 3.5 development environment to OpenROAD 2006.

Upgrading OpenROAD 3.5 to OpenROAD 2006 is straightforward, provided the upgrade is properly prepared for and carefully executed.

1 — Introduction

The recommendations in this document are based on extensive practical experience of such conversions, and have two objectives:

- To ensure that your upgrade is successful the first time
- To minimize time and effort required to upgrade

Upgrading OpenROAD 3.5 to the current OpenROAD 2006 is straightforward in most respects, provided it is properly prepared for and carefully executed. Much of the upgrade work involves the automated correction of known upgrade issues covered later in this document. In most other cases, minimal code changes should suffice. However, certain issues need careful attention, and one may require significant work (the move away from database drivers for installations that use non-Ingres databases). It is most unlikely that a completely manual upgrade will be either cost-effective or of sufficient quality.

This document is intended to be read in conjunction with the companion document "Upgrading OpenROAD Applications from 4.1 to 2006" (referred to in the text as the 4.1 to 2006 document). Together they define a recommended upgrade process for systems currently on OpenROAD 3.5.

While the companion document addresses the detailed logistics of a successful conversion, this document focuses exclusively on the known issues specific to upgrade from OpenROAD 3.5 that you may encounter, and their resolution.

2 — Summary of the Upgrade Process

The upgrade process consists of the following basic phases. Each phase is described in detail in the companion 4.1 to 2006 document, and is not considered further here.

Phase 1: Prepare for the upgrade. Freeze the codeline (apart from emergency fixes) as a final step.

Phase 2: Convert the applications, including resolving known issues.

Phase 3: Conduct acceptance testing.

Phase 4: Go live. Unfreeze the codeline.

3 — Known Issues

Ensure that any already identified, existing problems with your applications have been corrected or documented. More problems may appear during the conversion process; deal with each of them in one of these two ways before proceeding. Not correcting or documenting problems as they are identified is likely to waste considerable time during the conversion process.

Field Property-related Issues

These are the most common issues. Color and transparency issues are hard to detect and quantify comprehensively by manual means, and realistically they cannot be fixed manually, either, because there are far too many occurrences. But they can be corrected simply, using an automated tool.

Incorrect FP_CLEAR field setting in existing applications

This developer issue is widespread and will need to be corrected in most applications. It typically manifests as frames having areas with the wrong color. Ingres provides an automated PropertyChanger utility to assist you in correcting this problem. To request the PropertyChanger tool, contact Ingres Support at http://www.ingres.com/services/enterprise-support.php.

For more information about this issue and its resolution, see the 4.1 to 2006 companion document.

Color settings

If you are using your own color definition file for OpenROAD and plan to continue to do so, you may skip this section.

ISSUE: OpenROAD changed the color definitions it uses between version 3.5 and 4.0. These definitions have not changed since.

In applications developed initially in OpenROAD 3.*x*, CC_PALE_GRAY and CC_LIGHT_GRAY were identical, and many fields were set to one or other indiscriminately.

The same thing happened to CC_SYS_BTNFACE when Microsoft's definition of their button gray was switched from rgb(192,192,192) in Windows NT to rgb(212,208,200) for Windows XP.

When you convert to OpenROAD 2006, the grays will be unacceptably different and will need to be reconciled. In many cases, the other colors will differ from those in the OpenROAD 3.5 application potentially to such an extent that they also will need to be changed.

RESOLUTION: If you decide to reset the non-gray colors, the most effective and low-cost route is to define your own color table (see the following procedure).

We recommend that you reset the grays. Since no adjustment of the color table will deal with the change to CC_SYS_BTNFACE, and the distinction between CC_PALE_GRAY and CC_LIGHT_GRAY in existing applications is a developer issue and not an intentional implementation, we recommend that you use the PropertyChanger tool mentioned previously to correct this (see the 4.1 to 2006 companion document).

To reset the color table

1. Make a copy of the apped.ctb file in your OpenROAD 3.*x* installation (\ingres\files\apped.ctb). Name it something like apped2006.ctb.
2. Paste the file into the corresponding folder in your OpenROAD 2006 installation.
3. In a command window on your OpenROAD 2006 installation (that is, with the II_SYSTEM appropriate to that installation), issue the following command to point the OpenROAD color handler to your new file:
   ```
   ingsetenv II_COLORTABLE_FILE apped2006.ctb
   ```
4. Run your application from Workbench and confirm that the colors are correct. In the unlikely case that they are not, you will need to adjust the settings in the CTB file until the colors are acceptable.
5. Amend your deployment specifications to ensure that II_SYSTEM\files will contain apped2006.ctb, and II_COLORTABLE will point to apped2006.ctb in each upgraded client installation.

To correct the gray settings

See the 4.1 to 2006 companion document for details.

TableField pseudo-headers

If you are certain that all your TableFields use OpenROAD's built-in TableField header or use no TableField header, you can skip this section.

If your application may have some pseudo-headers built within a separate StackField, you will most likely need to make corrections.

It is possible that a simple font change will fix this problem in most or all places. Where it does not, you will need to scan the entire runtime application manually for table header alignment discrepancies, and correct them individually.

ISSUE: TableFields in OpenROAD 3.*x*, although powerful, had limited column header support compared to rival products. A number of application developments replaced the native OpenROAD TableField header by a more visually acceptable StackField-based pseudo-header, whose contents were manually adjusted to align with the TableField columns. In many cases, the application code included specific references to these pseudo-header fields.

Font-mapping changes in OpenROAD 4.0 and beyond cause virtually all such pseudo-headers to become misaligned unacceptably, and application changes will be necessary.

RESOLUTION: Redefining the TF_SYSTEM font in appedtt.ff (or the font file referenced by II_FONT_FILE, if defined), may resolve this problem. You can reset it to the font specified in the OpenROAD 3.5 installation (not recommended), or you can switch it to Arial. This solution will change all fields that currently use TF_SYSTEM; be sure that the result is acceptable.

To redefine the TF_SYSTEM font

1. Make a copy of the appedtt.ff file (or the font file referenced by II_FONT_FILE, if defined) in your OpenROAD 2006 installation (\ingres\files\appedtt.ff). Name it something like appedtt2006.ff.
2. Edit this file and replace the font name used in the 6 definition lines supplied for TF_SYSTEM with the name "arial". Save and close the file.
3. In a command window on your OpenROAD 2006 installation (that is, with the II_SYSTEM appropriate to that installation), issue the following command to point the OpenROAD color handler to your new file:
   ```
   ingsetenv II_FONT_FILE appedtt2006.ff
   ```
4. Run your application from Workbench. Check that the TableField pseudo-headers are correctly aligned and that the effect on other fields remains acceptable. If not, re-edit the appedtt2006.ff file and the corresponding file in your OpenROAD 3.*x* installation. Copy the 6 definition lines supplied for TF_SYSTEM from the 3.*x* file, and use them to replace the corresponding 6 lines in the 2006 file.
5. Re-run your application from Workbench. Check that the TableField pseudo-headers are correctly aligned and that the effect on other fields remains acceptable.

If redefining the TF_SYSTEM font does not resolve this problem, you will need to adjust the field widths in the pseudo-headers to realign the fields.

Identifying and manually adjusting pseudo-headers is painstaking and slow. Using an adapted crawler tool is undoubtedly more cost-effective, but it will need to be adapted and tested to match the pseudo-header implementation. In this situation we recommend that you engage Ingres Services to assist with the conversion.

Date-datatype Related Issues

The following issue is related to date-datatypes.

Timezone issue

This issue will affect you only if your application has time data from multiple timezones (typically timetables).

ISSUE: OpenROAD 3.*x* uses absolute timezone handling based on the Ingres 6.4 model; this has certain built-in inaccuracies, and later versions of Ingres (and OpenROAD 4.0 onward) use timezone information tables, which are accurate.

Consequently, OpenROAD 3.5 applications handling time data as described previously typically have workaround code to correct the client data (otherwise a French train leaves an hour before the English client application says it will).

RESOLUTION: Upgrades to OpenROAD 2006 must disable the workarounds wherever they occur in the code.

Code-related Issues

The following issues are related to code.

Nullable declarations

If your application code has been policed well and routinely declares scalar variables as not null unless a null value is absolutely necessary, you can skip this section.

If your code contains many instances where “not null” has been omitted from declarations, you should consider this issue carefully.

ISSUE: In all versions of OpenROAD, all scalar field variables, global variables, and userclass attributes are by default not null; however, variables declared in the code are by default nullable, in accordance with the ANSI standard.

Up to OpenROAD 3.5, statements where the value of a nullable variable was assigned to a not-null variable or attribute were tacitly tolerated, despite the fact that they could fail at runtime. Likewise, substitutions in SQL statements could be done using nullable variables, despite the risk of failure.

However, this practice is inconsistent with OpenROAD's core policy of highlighting the maximum number of potential errors at compile time, when they are easiest to diagnose and fix. Accordingly, from OpenROAD 4.0 onward, such statements generate a warning (or for SQL statements, an error) at compile time.

RESOLUTION: The most robust solution, which is to locate and change all scalar nullable declarations in the code, can be automated. (Ingres Services has such a tool.) However, if there are any dependencies in the code along the lines of "if this_variable is null then …", the code is broken; such dependencies are likely to exist in at least a few places.

You cannot simply correct the declarations of variables that throw a warning or error; in many cases the effect is simply to move the warning or error one step back in a chain of assignments, while widening the net of dependencies.

The only negligible-risk approach is to apply the ifnull function to every statement throwing a warning. This cannot be automated. (Many clients have chosen to tolerate the warnings that are created whenever they compile or build their applications rather than engage in large-scale correction.)

For SQL errors, slightly more extensive recoding is necessary and also manual.

To apply ifnull to statements that are causing a nullability warning, and to fix SQL statements that are causing a nullability error

In Workbench, correct each application , as follows:

1. Compile the application (with force compile set) and copy w4gl.log from ingres\files to *applicationname*.log. Open the log file. In Workbench, view the components of the application.
2. Correct each component that has compile warnings, as follows:
    a. Open the component and compile it with "ignore compile warnings" cleared.
    b. Use the compile-errors window to correct each warning statement, as follows:
        - Navigate to the warning, using Ctrl-W.
        - Open the editor at the offending codeline, using Ctrl-F7.
        - Check the declared datatype of the relevant variable (or expression).

- Apply ifnull according to the datatype. For example, if in the following statement "nullable_var" is of datatype varchar:

  ```
  notnull_var = nullable_var;
  ```

  change the statement to:

  ```
  notnull_var = IfNull(nullable_var, '');
  ```

- Navigate to the next warning using Ctrl-W, and repeat the procedure.
- Process all warning statements in this way. Recompile to ensure they are fixed.

3. Use the compile-errors window to correct each SQL error statement, as follows:

   a. Navigate to the error, using F3.

   b. Open the editor at the offending codeline, using Ctrl-F7.

   c. Add a code line before the error statement that copies the nullable variable, ifnulled, into a companion not null variable (same name, suffixed with '_nn', for example); then alter the error statement to use the notnull variable. For example, if the error statement is:

   ```
   UPDATE :table_name SET year = 1984;
   ```

   change the code at this point to:

   ```
   table_name_nn = IfNull(table_name,'');
   UPDATE :table_name_nn SET year = 1984;
   ```

   d. Declare the companion not null variable alongside the declaration of the offending variable.

   e. Navigate to the next error, using F3, and repeat the procedure.

   f. Process all error statements in this way. Recompile to ensure they are fixed.

4. Save and close the component.

5. Repeat with the next component until the application is clean.

6. Repeat with the other applications.

Database-related Issues

The following issues are related to the database.

Database drivers

If you are connecting to a non-Ingres database using one of the provided drivers, you will have to change your application to use the corresponding Ingres Enterprise Access component (these are used extensively by applications already running on OpenROAD 4.0 and above).

ISSUE: OpenROAD 4.0 and above provides no support for specific database drivers to non-Ingres DBMSs.

RESOLUTION: The Ingres Enterprise Access components provide specific, high-performance, high-functionality interfaces to a more complete list of DBMSs.

If you are using a database driver, Knowledge Base document 355675 on Ingres ServiceDesk provides detailed guidance about using Enterprise Access with OpenROAD.

Contact Ingres Support for additional guidance concerning how to upgrade your driver-based applications.

Platform-related Issues

The following issues are related to operating system platforms.

Cursession.WindowSystem

You need to consider this issue only if your application runs on both UNIX and Windows platforms, and uses CurSession.WindowSystem (rather than CurSession.OperatingSystem) to determine the operating system. The correction involves 3 lines of code.

ISSUE: In versions of OpenROAD since OpenROAD 4.0, the CurSession.WindowSystem setting for UNIX platforms is set to WI_WINN32, not WI_MOTIF, reflecting that OpenROAD is running on a pseudo-Windows environment—Mainsoft—rather than on UNIX itself, so that the features available to it will be "Windows" ones. Developers who want to determine whether they are hosted on UNIX use the CurSession.OperatingSystem setting to look for SY_UNIX.

However, some OpenROAD 3.5 applications deployed to both UNIX and Windows platforms have tested CurSession.WindowSystem for a WI_MOTIF setting rather than testing CurSession.OperatingSystem for SY_UNIX. This will not work when they are converted to OpenROAD 2006.

RESOLUTION: Because of this, in OpenROAD 2006 (but not OpenROAD 4.*x*) the CurSession.WindowSystem attribute is ReadWrite, and can be reset in the code from WI_WINNT32 to WI_MOTIF when the application starts. Thereafter, any code that tests CurSession.WindowSystem will distinguish a Windows host from a UNIX host.

To ensure that CurFrame.WindowSystem identifies UNIX operating systems as UNIX

1. In Workbench, open the starting component of your application and edit its script.
2. Immediately after the "begin" (or "{") statement of the inititalize or procedure block, insert the following 3 lines of code:

```
IF CurSession.OperatingSystem = SY_UNIX THEN
    CurSession.WindowSystem = WI_MOTIF;
ENDIF;
```

3. Save and close the component.

Fonts in mixed UNIX/Windows environments

If you want to have your converted applications display the same way (as closely as possible) on both UNIX and Windows, you will need to read this section.

ISSUE: The available UNIX fonts differ in slight but discernible respects from their Windows counterparts.

RESOLUTION: You can use II_FONT_FILE to ensure that your UNIX application uses the same font definitions as your Windows application, but the TF_SYSTEM fonts will not map well.

To display fonts on UNIX as closely as possible to the Windows display

1. Set II_FONT_FILE to point to the default font file (appedtt.ff), or the version of it that you are using on Windows.
2. Change the TF_SYSTEM setting in your choice of font file from "MS Sans Serif" (which maps poorly on UNIX) to "arial".

If you still experience problems, read Ingres Knowledge Document 272124 for further suggestions.

UNIX-specific Issues

The following issues are related specifically to UNIX platforms.

Fonts

If your 3.5 application specified UNIX-specific fonts, you will need to review and probably change your font file when you upgrade to 2006.

(Also see the "Fonts in mixed UNIX/Windows environments" in the previous section.)

ISSUE: The default fonts specified in the OpenROAD font file appedtt.ff are now Windows fonts.

RESOLUTION: You can overwrite these definitions, but exercise care when doing so.

To overwrite the font-file definitions

1. Make a copy of appedtt.ff, naming it something like appedtt2006.ff, and edit the file.
2. Replace the font names with the appropriate UNIX font name ("-*-lucida-medium-r-*-*-*-120-*" for example). Ensure that the order of the changed fonts is appropriate to the font sizes in the file.
3. Set II_FONT_FILE to point to your edited file.
4. If you already had your own font file in OpenROAD 3.5, review and change it to ensure that the order of the fonts exactly matches the appedtt.ff font order.

Log/Trace destination

If you want the trace output to continue to go to stdout, you must change the application command line.

ISSUE: Log/trace output no longer goes to stdout in OpenROAD 2006. Instead, there is a trace window and a trace w4gl.log file.

RESOLUTION: You can override the destination in the command line.

To override the trace output destination

1. In the command line, include the –L flag to redirect the log output.
2. To remove the trace window, include the logonly option in your –T flag setting.

For more information about command flags, see the OpenROAD 2006 *User Guide*.

3GL libraries

If you have 3GL libraries, you will need to change the way you build them.

ISSUE: make3gllib no longer exists.

RESOLUTION: In the OpenROAD installation II_SYSTEM/ingres/w4glsamp directory, there is a 3GL sample. Use this and the instructions provided to build your 3GL libraries in OpenROAD 2006.

WidgetId

Although its use is unsupported, some OpenROAD 3.5 applications pass the WidgetId in XWindows calls. If your application does this, you will need to make changes.

ISSUE: Now that OpenROAD on UNIX runs via MainWin, using the WidgetId in this way will no longer work without changes.

RESOLUTION: Open a call with Ingres ServiceDesk, as any solution will be different for different applications.

INGRES CORPORATION : 500 ARGUELLO STREET : REDWOOD CITY, CALIFORNIA 94063
PHONE +1.650.587.5500 : FAX +1.650.587.0000 : www.ingres.com : For more information, contact info@ingres.com