Devguide 3.0.1 Release Notes D

This appendix is a supplement to the following four OpenWindows Developer's Guide documents:

- *OpenWindows Developer's Guide: User's Guide* (this document)
- *OpenWindows Developer's Guide: XView Code Generator Programmer's Guide*
- *OpenWindows Developer's Guide: OLIT Code Generator Programmer's Guide*
- *OpenWindows Developer's Guide: Motif Conversion Utilities Guide*

Devguide 3.0.1 is the successor to Devguide 3.0, which is the immediate successor to Devguide 1.1. There was no Devguide 2.x release. The OPEN LOOK toolkit, XView, and the OPEN LOOK Intrinsics Toolkit (OLIT), provide different levels of OPEN LOOK compliance.The following subjects are discussed in this appendix:

- Compatibility issues regarding Devguide 3.0.1
- New features of Devguide 3.0.1
- Differences between Devguide versions 3.0 and 3.0.1
- Known problems with the Devguide software
- Features not supported by the code generators

Compatibility

Devguide 3.0.1 and its associated code generators (GXV, GXV++, and GOLIT) produce code compatible with the Solaris 2.x operating environment and the OpenWindows Version 3 environment.

D

Devguide 3.0.1 can read GIL files produced by Devguide 1.1 and 3.0 and convert them into Devguide 3.0.1 format. The XView code generators GXV and GXV++ can also read Devguide 1.1 GIL files and generate OpenWindows Version 3 code.

New Devguide Features

Code Generator Tool: Devguide's code generator tool generates and compiles code, and runs applications for the GXV and GOLIT code generators, without requiring you to type commands at the command line. The code generator tool's main window is resizeable.The inner pane is a command tool.

Default Generated Code: Devguide now produces ANSI C as the default version of C. This means that procedure names and prototypes have their parameters listed in the declaration. For example:

```
void
my_callback(Panel_item item, Event *event)
{
    foo_window1_objects *ip = (foo_window1_objects *)
xv_get(item, XV_KEY_DATA, INSTANCE);

    fputs("foo: my_callback\n", stderr);

    /* gxv_start_connections DO NOT EDIT THIS SECTION */

    xv_set(Foo_window1->button2, XV_SHOW, TRUE, NULL);

    /* gxv_end_connections */
}
```

Devguide 3.0, by default, produced K&R C code such as this:

```
void
my_callback(item, event)
    Panel_item     item;
    Event          *event;
{
    foo_window1_objects *ip = (foo_window1_objects *)
xv_get(item, XV_KEY_DATA, INSTANCE);

    fputs("foo: my_callback\n", stderr);

    /* gxv_start_connections DO NOT EDIT THIS SECTION */

    xv_set(Foo_window1->button2, XV_SHOW, TRUE, NULL);

    /* gxv_end_connections */

}
```

Clearly, these forms are different. As a result, if your application is built with Devguide 3.0, you might experience problems merging the `_stubs.c` file(s). Also, if you use GXV and you did not use the -a switch parameter when you generated code, the 3.0.1 compiler will produce unexpected results when you compile your 3.0 application.

See Appendix B, “Porting Devguide 3.0 Applications to Devguide 3.0.1," in this manual for details.

Access From Within SPARCworks or ProWorks: If you are using SPARCworks or ProWorks, and you want to access Devguide through the SPARCworks or ProWorks manager, you can add Devguide to the manager as you would any tool. After you do this, the Devguide icon appears on the SPARCworks or ProWorks palette whenever you start up SPARCworks or ProWorks. You do not have to leave the SPARCworks or ProWorks context to use Devguide.

See Appendix C, “Adding Devguide to the SPARCworks or ProWorks Environment,” in this manual, for more information. The method described in Appendix C for adding Devguide to the SPARCworks or ProWorks manager is effective only for the current session of the manager. If you quit and restart the manager, the Devguide icon no longer appears on the SPARCworks or ProWorks palette. To permanently add Devguide to the SPARCworks or ProWorks manager, do the following:

1. **Copy the `sparcworks.defaults` or proworks.defaults file from the `lib` subdirectory where SPARCworks or ProWorks is installed. For example, if SPARCworks is installed in `/opt/SUNWste/SW2.0`, you would type the following:**

```
% cd
% cp /opt/SUNWste/SW2.0/lib/sparcworks.defaults .
% chmod +w sparcworks.defaults
```

2. **Edit the `sparcworks.defaults` or proworks.defaults file and make the following modifications:**
   - Add "devguide" (no quotes) to the end of the `swToolBox.tools` line.
   - Add the following to the bottom of the file:

```
swToolBox.applications.name: devguide
swToolBox.applications.glyph.label: Devguide
swToolBox.applications.glyph.file:
/opt/SUNWguide/include/images/guide_32X32.icon
swToolBox.applications.command: devguide $SUNPRO_SWM_TT_ARGS
$SUNPRO_SWM_GUI_ARGS
```

   - Modify the path to the icon file so it matches the path to where you have Devguide installed.

3. **Make sure that you have the Devguide `bin` directory in your path.**

4. **Start SPARCworks or ProWorks.**
   The Devguide icon should appear on the SPARCworks or ProWorks pallette.

Note – If SPARCworks or ProWorks generates an error message with reference to the `alert_qmark32.pr` icon, edit the `sparcworks.defaults` or proworks.defaults file to read:
`/usr/openwin/include/images/alert_qmark32.pr`

New GXV Features

GXV provides two new command line options (or flags) to enhance or customize code generation:

- `-k` (`-kandr`) —Specifies that K&R C code be generated (ANSI C is now the default).
- `-sb` —Adds the `-xsb` switch to the compiler options in the Makefile if you are using ANSI C. Adds the `-sb` switch if you are using K&R C. The resultant compiled code can then be used with Source Browser in SPARCworks.

See the *OpenWindows Developer's Guide: XView Code Generator Programmer's Guide* for complete information on these and other GXV command line options.

New GOLIT Features

GOLIT provides the following new command line options (or flags) for code generation:

- `-i` (`-international`) —Writes all Level 3 internationalization-specific resources into a resource file.
- `-k` (`-kandr`) —Writes K&R C code that doesn't contain function prototypes. Golit produces ANSI C code by default.
- `-r` (`-resources`) —Writes all resources into a resource file.
- `-sb` —Adds the `-xsb` switch to the compiler options in the Makefile if you are using ANSI C. Adds the `-sb` switch if you are using K&R C. The resultant compiled code can then be used with Source Browser in SPARCworks.

See the *OpenWindows Developer's Guide: OLIT Code Generator Programmer's Guide* for complete information on these and other Golit command line options.

Devguide Changes From the Previous Release

Drag and Drop: In Devguide 3.0, a NOTIFY_SAFE interposer was placed on control areas and canvases that contained items or could themselves be the destination of a drop operation. Devguide 3.0.1 places a NOTIFY_IMMEDIATE as well as a NOTIFY_SAFE interposer on these objects.

D

GNT Changes From the Previous Release

The GNT (Guide to TNT) code generator, along with all pertinent GNT manuals, is now located in the demo directory. This means that GNT is not a fully supported code generator, and is instead provided as a demonstration code generator for the TNT (NeWS) toolkit.

GNT will not be enhanced in future releases of Devguide, and will be removed from the demo directory in the next release of Devguide. Those developers who have used GNT in the past should plan to migrate their work to other toolkits using the GXV (Guide to XView) or the GOLIT (Guide to OLIT) code generators.

Known Problems with the Devguide Software

Drag and Drop Targets: If you create a drag and drop target in your application, create a CallFunction connection with DroppedUpon as the When item. Select the drag and drop target as both Source and Target for the connection. If you do not create this connection, the drag and drop target will appear to accept drops from File Manager, even though no file transfer occurs. In this case, the drag and drop target will send a delete request to File Manager. This causes File Manager to delete the file, even though the drag and drop operation was unsuccessful. This behavior results from an XView bug.

In addition, you cannot drag and drop within the same application. That is, the source and destination can't be within the same application. This is not so much a drag and drop limitation as it is a Devguide gdd library problem.

Groups: When saving, Devguide may cause a core dump on a GIL file with a large number of groups. This occurs only on rare occasions. If it does happen, however, restart Devguide, ungroup one of the larger files (or the largest), and then save again.

Groups and Anchors: If you create a GIL file containing a group and anchor object, and the group is positioned above the object that you want to anchor it to, Devguide ignores the anchor object. That is, the group is not anchored to the object in this case.

Sharing Menus: If you create a menu that is shared by objects in more than one .G file, put the menu in a .G file separate from those containing the object descriptions. Load the .G file containing the menu object description whenever you load any .G file that references the shared menu. Menu references in any file that does not contain the menu description are lost.

Known Problems with the GXV and GXV++ Software

GXV Merging: The GXV merging facility will not catch all programmer errors. In particular, if you forget to close off a quoted string (that is, you only put in the first double quote) in your `_stubs.c` file, the code generated will not contain any of your changes. The `_stubs.c.BAK` file, however, retains your old version. If you have any doubts about a `_stubs.c` file, check your `.BAK` file, or copy your `_stubs.c.BAK` to `_stubs.c` and fix your code.

GXV and GOLIT Stubs File Merging: Merging the `*_stubs.c` files on successive .G file processing does not work for all valid C or C++ syntax. For example, `char * ptr = dgettext("Text1","Text2");` where `ptr` is global, produces the following error:

```
<codegen> cannot parse <file> beyond lib <line>
```

This construct does not cause an error, for example, within `main() {}`. To allow the file to be merged, comment out the offending lines before running GXV or GOLIT. Then uncomment the lines before compiling.

XVPS Library: GXV and GXV++ use the XVPS library to implement the PostScript drawing option for canvas panes. The XVPS library is an XView extension that is included with the OpenWindows Version 2 and Version 3 environments only as a demo. XVPS and the PostScript drawing model option are not supported by SunSoft.

Note – Applications should *not* rely on XVPS's future existence *nor* should they rely on future versions of XVPS being binary compatible with present versions.

Multiline Text Field Display: Due to an XView bug, multiline text fields do not display correctly when their parent control areas are displayed. This impacts applications in which control areas are made visible and invisible—for example, when using layered panes.

To remedy the problem, whenever your application uses XV_SHOW to display (hide) a control area you must insert code using XV_SHOW to display (hide) the multiline text field.

Help -? Option: The -? option does not work properly if run in the C shell. The ? is interpreted by the C shell, and is not passed on to gxv or gxv++. If you use the C shell, and you want to view a list of all GXV (GXV++) command line options, use the -help option instead.

Generating Help Files for Projects: You cannot generate help files for a project by using the -h option in combination with the -p option. The commands gxv -h -p my_project and gxv -p my_project produce the same results. They generate files associated with the .P file my_project. No help files are generated in either case. If you want to generate help files for a set of GIL files in a project, you mustgenerate the help files individually. Run gxv on each GIL file using the -h option.

Known Problems with GNT Software

Initial Values and Initial States: You must include application code to set initial states of menu buttons, such as Active/Inactive or Selected.

Groups: Groups create excess overhead and their use in GNT is not recommended.

Internationalization: Internationalization is not fully supported by TNT in OpenWindows Version 3. Window menus are not localizable in this release.

Localization: During localization, after the interface is frozen, do not group and ungroup buttons. Grouping and ungrouping buttons changes the number of tokens created. If you group or ungroup buttons during localization, the executable will no longer run because the number of tokens will no longer match the number of buttons.

Menus: Create a menu before you attach it to another object. Otherwise, the menu is referred to in the GIL file before it is defined, and you will get a PostScript undefined error. Alternatively, you can edit the GIL file so that the menus are defined before they are referred to.

Unresolved Installation Bugs

Problem: there are no links for doc and man after installation.

Workaround: Do the following to create the links:

```
% cd /opt/SUNWguide [or your $GUIDEHOME, if different]
% su
Password: [enter root password]
# ln -s share/man man
# ln -s share/doc doc
```

Unresolved Devguide Bugs

The currently outstanding bugs for Devguide (listed according to bug ID number) are as follows:

1071386: Borders for text panes are not shown while building an interface within Devguide. Generated applications layout will be somewhat different.
1086058: There is no way to specify the spacing between choices in checkbox settings.
1060374: An anchor object is not displayed on the Groups Properties window when loading a GIL file.
1091778: Can't use something like `foo::myfunc` for a call function.
1081031: Choosing Save As... with the Project Managers Project button doesn't save .G files.
1097702: Devguide doesn't preserve symbolic links to GIL files correctly.
1080931: Devguide mishandles newline in `PANEL_LABEL_STRING`.
1079760: Stubsmerge doesn't handle unpaired quotes well.
1077355: Can't remove unwanted connections from the .P file with Devguide.
1072360: Devguide doesn't support all create-only attributes of XVIEW.
1082534: `Printfs` are being placed inside the `gxv_start_connections` area.
1085133: Panel size in GXV generated code is different from Devguide prototype.

Unresolved GXV Bugs

1093669: GXV generates bad event handler for `PANEL_TEXT`.
1097049: GXV cannot parse function prototypes.
1096076: `gxv -h` does not work in conjunction with projects.
1095825: GXV generates code that does not compile when using ExecuteCode.
1078451: Code generation for setting item fails if "when" is deselected.

D

1082698: Menu handles aren't accessible to stubs file.
1087522: GXV cannot handle multi-level indirection of functions.
1081030: GXV++/stubsmerge doesn't deal with argument defaults well.
1099391: Can't use functions declared in `gfm_ui.h` from C++ in SVR4.
1088170: Dragging from a drop target to a scrolling list on the same panel doesn't work.
1082374: Colors are incorrect when used on MacX.

Unresolved GOLIT Bugs

1099852: Scrolling list items placed inside a scrolling list in Devguide don't show up in a GOLIT generated application.
1079909: There is no callback when a drag and drop from source to target is used.
1091684: GOLIT doesn't create multi-line text widget
1091376 OLIT canvas created via GOLIT doesn't resize properly
1113998 Null pointer passed to callback function; occasionally, for connections across interface files or across windows, code is generated that uses a null pointer within a callback.

Unresolved GUIL Bugs

1154709: GUIL truncates two characters of file name if you do not include .G or .P suffix when you execute command. WORKAROUND: include suffix.

GOLIT Features not Implemented in the Current Release

Bold Labels: The OPEN LOOK specification calls for the labels of all control objects to be bold. When you prototype an interface in Devguide, the label font will appear bold. However, the default font weight for labels in GOLIT-generated interfaces is not bold. When you run your GOLIT program, the labels will not be bold. You can set the caption font resource by setting the fallback resources in the `_stubs.c` (or `<project>.c`) file, or application defaults file. You can achieve the same effect by setting the caption font resource on the command line in the following manner:

```
% my_application -xrm '*Caption.font: lucidasans-bold'
```

where *my_application* is the name of the application for which you want the labels to appear bold.

Help: GOLIT does not support Help as implemented in Devguide. To add help for a widget, you must use a Create Proc and `OlRegisterHelp`(). For more information, see Chapter 4 of the *OpenWindows Developer's Guide: OLIT Code Generator Programmer's Guide*.

Miscellaneous Unsupported UI Elements: The following UI elements are available in Devguide but are not currently supported by GOLIT:

- Glyph types in labels
  Chapter 4 of the *OpenWindows Developer's Guide: OLIT Code Generator Programmer's Guide* provides an example of how you can use Create Procs to display any bitmap image in a label.
- Icons for Base Windows
  Chapter 4 of the *OpenWindows Developer's Guide: OLIT Code Generator Programmer's Guide* provides an example of how you can use Create Procs to assign an icon to your application.
- Term Panes
- Numeric Text Fields
- Drag and Drop from a Canvas or a Scrolling List
- List items
  The scrolling list itself is supported. However, GOLIT omits any list items that you inserted in Devguide. You must write your own code to insert list items.
- Setting stack
  GOLIT provides limited support for this object. The current setting is not displayed next to the abbreviated menu button.
- Layered panes in pop-up windows
  If your interface has layered panes in a pop-up, the panes will be tiled in your compiled application. Layered panes do work in base windows.
- Slider end values
  Although you can include slider end values in your interface prototype in Devguide, they will not appear in your compiled application.

GOLIT simply ignores most of these unsupported elements when it reads the GIL file. However, if you attempt to use GOLIT to generate code for an interface that includes term panes or numeric Text Fields, GOLIT will abort code generation and display an error message.

D

GNT Features that are not Implemented in the Current Release

Connections: Some actions (connections) have not been fully implemented. An error message appears:

```
Undefined action for <object type> <name> What code <number> Action
<Action Name>
```

Miscellaneous Unsupported UI Elements: The following user interface elements are available in Devguide but are currently not supported by GNT.

- Term panes
- Multiline text fields
  Multiline text fields are converted to single line text fields.

Internationalization: GNT supports string translation, but does not support the following:

- Window menu does not reflect LOCALE
- Decimal delimiter is not changed for LOCALE

Layered Panes: You can layer only control areas in GNT.

Do not use a canvas object as a member of a layered pane; you will not be able to display the canvas because the Show action does not support canvas objects. When you use a canvas with a Show action, GNT issues the following warning and then exits.

```
{31} gnt test
Running Postscript Code Generator
Undefined action for Canvas gtst_window1_canvas1 What code 4
Action Show
```

GNT does not support layering text panes. When you use a text pane in a layer of panes, GNT issues the following warning and then exits.

```
{31} gnt test
Running Postscript Code Generator
Stacked text panes are not supported by gnt
```