LabVIEW™ 5.1.1 Release Notes

These release notes describe the changes to LabVIEW and the process of upgrading LabVIEW to version 5.1.1. Refer to the *Required System Configuration* section of the *LabVIEW Version 5.1 Addendum* before you intstall LabVIEW 5.1.1.

How to Upgrade

Upgrading LabVIEW is an automated process. When you open a VI that was created with a previous version, LabVIEW automatically converts and compiles the VI.

Conversion is a memory-intensive operation. When LabVIEW loads a VI that was saved with an earlier version, it loads all components of the converted VI (front panel, block diagram, data) into memory and then compiles the VI in memory. In addition, LabVIEW loads the components of all subVIs needing conversion into memory.

You can estimate how much memory LabVIEW needs to convert VIs by adding the amount of memory your VIs and all their subVIs occupy on disk. If they are in VI libraries, you should add approximately 30 percent of the VI library size because the VIs are compressed. The conversion process might require at least that much memory plus an additional 3MB of memory to run LabVIEW.

If your computer does not have enough memory for you to convert your VIs all at once, convert the VIs in stages, by components. Examine your hierarchy of VIs and begin by loading and saving subVIs in the lower levels of the hierarchy. You can then gradually work up to the higher levels of the hierarchy. You also can select **File»Mass Compile** to convert a directory of VIs. Notice, however, that this option converts VIs in a directory or VI library in alphabetical order. If a high-level VI is encountered first, **Mass Compile** requires approximately the same amount of memory as if you opened the high-level VI first.

You can monitor your memory usage using the **Help»About LabVIEW** option, which summarizes the amount of memory you have used.

If You Are Upgrading from LabVIEW 5.1

LabVIEW 5.1.1 contains no new features and fixes some bugs. There are only a few compatibility issues between LabVIEW 5.1 and LabVIEW 5.1.1 within the Application Builder.

Changes in Version 5.1.1

LabVIEW 5.1.1 corrects the following issues:

- There are numerous Application Builder changes. Refer to the *LabVIEW 5.1.1 Application Builder Release Notes* for more information about new features.
- Installation problems with Japanese LabVIEW 5.1J
- **Save With Options** errors that affected LabVIEW 5.1
- Error that crashed LabVIEW when renaming a knob control while the Show Help window was open
- The **Preferences»Path»VI Search Path** did not correctly recognize the path to VIs.
- The Peak Detector VI sometimes missed peaks at certain widths.
- In BridgeVIEW 2.1, VI Profiler failed to work.
- Performing an Undo by pressing <Control-Z> when using an array with **Update Value While Typing** caused LabVIEW to crash.
- LabVIEW bypassed the print spooler with certain printers.
- Run-time menus retained links after they were deleted.
- **Scale all objects on panel** did not work when calling a subVI after reducing screen resolution.
- LabVIEW crashed when saving a VI without its block diagram when using custom **Save with Options**.
- Under certain circumstances, Boolean indicators that have constants wired to them did not update properly.
- LabVIEW crashed if you used the **Add item after** option to add items to menu ring controls.
- Printing with Print Panel method in VI Server behaved inconsistently dependent on multithreading.
- If multithreading was disabled in LabVIEW and you attempted to run a VI that calls a subVI that was set to have a subroutine priority, LabVIEW crashed once the main VI completed execution.
- The Open Config Data VI did not have the error cluster from the File Info subVI.

- LabVIEW crashed under certain circumstances with the From Exponential/Frac/Eng function.
- LabVIEW produced an error message when a property of an ActiveX control was set to the refnum of an ActiveX Server object.
- LabVIEW 5.1.1 on Windows NT has increased memory usage from 5.1 to 5.1f1.
- A VI saved with development distribution options could not be opened in real time.
- If you used the **Edit Position** attribute for a table control, the selected characters in a cell lost focus.
- There was a significant performance lag on PowerMac when a parallel loop waited on an occurrence.
- The Application Builder encountered problems with DLLs referenced by name.
- LabVIEW threads would hang when trying to close an automation handle if the thread that created the handle was in a message loop that was not checking for the thread message that told it to close its handle.
- **To G Data** gave this error when converting a variant that contains an empty array of strings to a LabVIEW array of strings. LabVIEW error: `Error #3 : "memory.c", line 445` when **To G Data** is called.
- Assemble Test VI did not work under Windows 2000 with LabVIEW.
- Clusters did not update correctly when using the Bundle By Name function.
- In certain circumstances, multiple shift registers used in conjunction with the unbundle primitive could generate incorrect values.
- The Scan From String function did not return correct defaults when the default values were unwired.
- The **Auto-Center** feature did not work when the **Auto-Hide** feature was enabled.
- LabVIEW displayed an error if you tried to print a control with a strict type definition.
- Using relative paths for the `RTM` path did not work when importing strings to a LabVIEW VI.
- On a multithreaded version of LabVIEW, the **Make Current Values Default** method worked only if the VI or Global VI front panel was open and did not work on hidden controls.
- LabVIEW did not process empty optional parameters that were non-variants when calling an ActiveX Server.