SNiFF+™

Version 3.2.1 for Unix and Windows

Release Notes

WindRiver®

TakeFive Software, Inc.
A Wind River Company
San Jose, CA
E-mail: info@takefive.com

TakeFive Software GmbH
A Wind River Company
5020 Salzburg, Austria
E-mail: info@takefive.co.at

Copyright

Parts of SNiFF+:

Trademarks

Credits

The first version of Sniff was developed at the Informatics Laboratory of the Union Bank of Switzerland. Its development was considerably facilitated by the public domain application framework ET++.

Authors of the first version:

Walter Bischofberger (Sniff)

Erich Gamma (Sniffgdb)

Erich Gamma and André Weinand (ET++)

Table of Contents

SNiFF+ 3.2.1 Release Notes

SNiFF+ 3.2.1: List of Enhancements

- Integration of Tornado II on Unix and Windows

  You can find online documentation under **Help(?) > Application Papers**; PostScript/PDF documentation is available in `SNIFF_DIR/integrations/TornadoII/doc`
- pRISM+ 2.0 integration on Windows
- Windows 2000 supported
- A debugger adaptor for ObjectAda is now available.
- The Java Debugger now works with JDK 1.3.
- There are now two new requests in sniffaccess:

  `generate_recursive_make_dirs` and

  `generate_include_directives`

SNiFF+ 3.2: List of Enhancements since SNiFF+ 3.0.x

New Rapid Reference Technology

- SNiFF+ now provides an alternative cross reference technology - database-driven cross referencing. Under this system, X-Ref information is maintained in a database in the file system. This database is directly queried and only the relevant information is loaded to memory, resulting in constantly fast queries and negligible additional resource consumption.

Improved Java Support

- New SNiFF+ integration with the VisaJ Java GUI Builder. For details, please see the “VisaJ for SNiFF+J” documentation in `SNIFF_DIR/Visaj/doc/userguide`/
- Speed of Java compilation has improved significantly.
- The Java Debugger now works on all platforms that support JDK 1.1.x, except Unixware.
- A new version of the Java debugger which is based on the JDI interface is now available in JDK 1.2.x or newer. This debugger works only on Windows and Solaris. For details, please refer to the `jdijdb_debugger.pdf/.ps` file in `SNIFF_DIR/doc`
- SNiFF+ now generates dependencies for Java. As a result, Update Makefiles may be slightly slower than before.

Reworked Make Support

- A new reworked Make Support solution is now available making it easier to use.
- Speed of build has been significantly enhanced.
- Makefiles are now updated much faster.

General

- SNiFF+ 3.2 provides a new parser for the Functional Structure Definition (FSD) language. This language is used to develop the CATIA User interface. CATIA is a CAD/CAM tool developed by Dassault Systems.
- Templates, for all supported languages and for MFC, are provided in your `SNiFF_DIR/config/project` directory.
- SNiFF+ for Linux now supports `glibc` 2.1 based systems like RedHat 6.1, SuSe 6.2 or Caldera 2.2.x. Due to incompatibilities in `glibc`, older systems running `glibc` 2.0.x (e.g., RedHat 5.3, SuSe 6.0) cannot be used.
- The Retriever is now much faster for large projects due to an indexed search algorithm.
- The free parsers for Perl, Assembler (68 & 86) and shell scripts (`.sh`) are installed with the SNiFF+ core package.
- We now support the version control tool called TeamConnection 3.0.
- There is a new Welcome dialog that explains how to evaluate SNiFF+.
- There is a new Personal mode that allows you to use SNiFF+ with projects containing up to 200 files.
- SNiFF+ is now more Windows GUI compliant. Click-sorting in grid views is possible in the Project Editor, Symbol Browser and the Class Browser. Toolbars can be shown/hidden in all tools.
- All lists in the Project Editor, Symbol Browser, Class Browser and Configuration Manager can be copied to the clipboard. Multiple selection is also possible.
- Symbol sharing for ClearCase users now possible.
- New Microsoft HTML Help is available on Windows.
- It is now possible to save cross reference filter settings.
- You now only need one script to perform unattended updates of working environments. Please refer to the User’s Guide for information on how to do so.

- The SNiFF+ Parser Developer Kit is now packaged with SNiFF+ free of charge. On windows, you can install the parseapi package which is part of your SNiFF+ installation. On Unix, it is automatically installed. The SNiFF+ Parser Developer Kit libraries can be found in:

  On Windows:

  `SNiFF_DIR/parser_api/ParserInterface.a`

  On Unix:
  `SNiFF_DIR/parser_api/lib.$PLATFORM/ParserInterface.a`
  For more information, please refer to the SNiFF+ Parser Development Kit documentation in the SNiFF+ online documentation.
- The **Synchronize Checkmarked Projects...** command in the Project Menu of The Project Editor is much faster than before. However, files in your PWE with the same version as those in your SSWE or RWE (if you don’t have a SSWE) are not deleted. To delete these files, use the **Synchronize File...** command in the File menu. Please note that this command is slower than the first.
- When using the **Check Obsolete Files...** command in the Project menu of the Project Editor, you can now specify filenames and directories that should be ignored, when reporting obsolete files. For information on how to do so, please refer to the

  `IgnoreObsolete` file in your `SNIFF_DIR/config` directory.

Minor Enhancements

- We now have a new debugger adaptor for the native Unixware 7 debugger.
- The Cross Referencer has been extended to partially support object-oriented cross referencing; this option can be enabled in the Cross Referencer's X-Ref Filter dialog. This means that it takes dynamic binding into account for Refers To and Referred By queries for virtual methods (non final in Java). Refers To now also shows all methods overriding an invoked method. Referred By now also shows all locations where an overridden member function in a base class is invoked.
- The default layout for the Cross Referencer, Include Browser and Hierarchy Browser has been changed to give a better overview.
- There are two new buttons in the Filter dialog in the Cross Referencer, **Save** saves the filter settings as default and **Load** loads the stored default settings.
- Within SNiFF+, the Java debugger is now available by default for JDK 1.1.x with JFC (Java Foundation Classes) instead of Bongo. In this way, the Java debugger will be available for all platforms.
- The path information for loaded files in the DiffMerge tool is now displayed in a tooltip.
- You can now navigate directly to header files from current source files. To do so, position the cursor into the header file name in the Source Editor and choose the **Show File <***header file***>** command in the **Show** menu. Please note that this menu command only works for header files that are part of the project but not for header files that are outside a project.
- All available Windows fonts are now also available in SNiFF+.
- More Motif like menu handling on Unix. When you click on a menu, the menu remains open even when the mouse is released.
- It is now possible to customize the appearance of syntax highlighting in the Source Editor view of the Preferences.
- There is a new **Print** option in the DiffMerge tool. By default the print orientation is set to Landscape so that the differences can be easily read.
- By default, the **Use Cache** checkbox in the Open Project dialog is not selected. As a result, the status of files that have been changed outside SNiFF+ is also correctly reflected.
- It is now possible to stop the project generation process when generating new projects.
- The `rlogin` command is used for remote debugging instead of the `rsh` command. In this way many dbx debuggers can run correctly.
- Two **History** drop-downs have been added to the right of the Find and Change fields in the Find/Change dialog.
- A **Scroll Time** option has been added in the Others view of the Preferences. By default, it is set to 1 which is the maximum scrolling speed. To make it slower, increase the value in the range of 1 to 500.

- A new **Extended Symtab API Positioning** checkbox has been added in the New Project Setup view of the Preferences. When enabled, parameter names and the start and end positions of the argument list and the constructor initialization list are added to the Symbol Table. To activate this option, execute the **Force Reparse** command in the Project Editor.
- An **Additional** field has been added in the Project Targets view of the Project Attributes dialog. Thus it is now possible for you to separately enter the additional include directives without them being overwritten when you press the **Generate** button.
- On Unix, FLEXlm binaries can be separately installed per platform.
- There is a new **Project > Update Cross Reference Info** command in the Project Editor. This allows updating of Cross-Reference information without reparsing symbol information. Subsequent Referred By queries are then faster.
- There is a new **Project > Add New Subproject...** command in the Project Editor. This allows creating and immediately adding subprojects to selected projects.
- The following menu commands in the Project Editor have been renamed:

New name in Project menu	Old name
Load/Update Symbol Info	Load/Update Symbol Table
Delete Symbol Information	Delete Symbol Files
Synchronize Checkmarked Projects	Update Workspace...
Synchronize File Status	Update File Status

New name in File menu	Old name
Synchronize File...	Update File...

Hints, limitations and known problems in SNiFF+ 3.2.x

General

- In the SNiFF+ Working Environments tool, **adm PWE:For Absolute Projects** is written next to the Root node and appears after absolute project names in the Launch Pad. This can be safely ignored since this Private Working Environment (PWE) is only relevant when working with the Cross Reference Database.
- If you have problems opening the SNiFF+ online documentation on Windows, run `HHUPD.EXE` (the latest version of Microsoft HTML Help). This executable can be found on your SNiFF+ CD-ROM in the `HTMLHELP` directory or on our ftp site.
- Closing a project via sniffaccess or the cross reference database remote shutdown mechanism only works correctly when no modal windows are open.
- SNiFF+ does not support black and white terminals.
- Update scripts (e.g., `updateWS.sh`) on Windows work with forward slashes only.
- When using SNiFF+ with ClearCase, we suggest that you store your symtab files someplace outside the ClearCase file system. By doing so, SNiFF+ is able to extract symbol information faster than before.
- **On Windows**, when using SNiFF+ with ClearCase, note that clearmake cannot be run in the standard bash shell. Please contact TakeFive Support for advice on how to solve this problem.

  On Windows, to avoid "`bad command`" messages when running clearmake, do one of the following:

  - when running clearmake, enter `MAKE_PARAMS=`

  A space should follow `MAKE_PARAMS=`
  or

  - in the Platform Makefile, in the `MAKE_PARAMS= --no-print-directory` option, remove `--no-print-directory`

  On Unix and on Windows: you must use the option

  ```
  clearmake -C gnu
  ```

  to use clearmake successfully.
- For SNiFF+ 3.2.x, you need the latest SNiFF+ Parser Development Kit which is delivered with SNiFF+ 3.2.x, i.e., you cannot use the Parser Development Kit of SNiFF+ 3.0 or 3.1 with SNiFF+ 3.2.x. Recompile your old parsers with the new Parser Development Kit in order to use your old parser.
- In Source Editor, it can be that the toolbar icons are sometimes activated when they shouldn't be and vice versa.

- SNiFF+ does not support file systems which are limited to 8.3 file naming. CAUTION: SNiFF+ for Windows supports and makes use of long filenames (filenames > 8.3). You should not install SNiFF+ or store SNiFF+ projects on file systems that don't support long filenames (e.g., older releases of Novell Netware don't support long filenames).
- The recommended minimum monitor resolution for working with SNiFF+ is 800x600.
- RCS 5.7 and ClearCase on Windows do not accept blanks in user names.
- The **Attach** command in the Execution menu of the Debugger tool is not supported on Linux platforms.
- To access the man pages for FLEXlm modify the environment variable `MANPATH` like this:

```
setenv MANPATH $MANPATH\:$SNIFF_DIR/man          (for csh)

MANPATH=$MANPATH:$SNIFF_DIR/man; export MAN-    (for sh or ksh)
PATH
```

- To make sure that SNiFF+ can access the GNU tools installed by `install.kit` make sure that the PATH environment variable includes a path to `/usr/local/bin/`.
- When you load large projects, the following error message may appear:

  ```
  Error during read of project file.
  ```

  This is caused when your system reaches its file descriptor limit. Ask your system administrator to increase this limit.
- When you update the symbol information for large projects, we suggest that you don't select the **Force Reparse** command in the Project menu of the Project Editor.

  Instead, select all projects in the Project Tree of the Project Editor and select **Delete Symbol Information** from the Project menu, close the project and open it again. By doing so updating symbol information will be much faster.
- If you intend to modify the cross platform VCS Adaptors, we suggest that you don't modify the python files for these adaptors. Instead we supply the SNiFF+ 2.4 shell script adaptors which can be easily modified. For details, please refer to the Readme file in the `$SNIFF_DIR/old_vcs/` directory.
- The SNiFF+ executable, `sniff -v` does not print the version number and the date on Windows. Instead, the version number is printed to standard error.
- On several Intel based Unix systems the kernel is preconfigured to a maximum process size of 16MB. This can cause problems because SNiFF+ can easily reach this limit when loading projects. To avoid this we recommend to increase the limit to at least 32MB.
- When a subproject including a not already loaded language is added to a project the first time, the new language cannot be selected immediately in the language selector. The new language is available only after re-loading of the project.

Configuring SNiFF+ with Exceed (5.x and 6.x)

When using SNiFF+ with Exceed, all fonts are mapped to a standard text font ("system"). Thus you cannot see any bold or italic fonts in SNiFF+. To change this, do the following:

1. Open your **Xconfig** tool.
2. In the Xconfig tool, double-click on **Font**.
3. In the Font Settings dialog that appears, click **Font Database...**
4. In the Font Database dialog that appears, deselect the **Automatic Font Substitution** checkbox.
5. Press **OK**, **Close** and close Xconfig.

In addition, we recommend that you use 75dpi fonts. If these fonts aren't in the Exceed Font Database, ask your System Administrator to install them for you.

> **Note**
>
> Depending on the Keyboard you have, some keyboard shortcuts don't work when using Exceed. Characters addressed with `<Alt Gr>` combinations may not be interpreted correctly.

Using WinSock2 on Windows95

Using certain versions of WinSock2 for Windows95 can cause SNiFF+ to respond extremely slowly to keyboard events, e.g. typing something in the Source Editor. For instance, many versions of the Sun JDK for Windows include this version of WinSock2. If you experience this problem, please download the latest version of WinSock2 for Windows95 from the Microsoft WWW site at

http://www.microsoft.com/windows95/downloads/contents/wuadmintools/s_wunetworkingtools/w95sockets2/default.asp.

This version does not show this behavior.

Bash Shell Performance on Windows

The bash and processes using the bash e.g., *make* started by **Target > Make** or commands like `cd c:` may execute slowly. The reason for this is that SNiFF+ uses paths including a drive letter e.g., `D:/MyProject` while the bash natively only understands POSIX-like paths e.g., `//D/MyProject`. Bash can only quickly convert such paths if they are known in its internal mount table. This mount table is persistently stored in the Windows Registry and is only used inside bash for translating paths, so using it is risk free.

Workaround: SNiFF+ provides shell scripts which mount and respectively unmount Windows drives and directories which are important for bash like `/tmp` and `/bin`. These scripts are located in `SNiFF_DIR/bin/`.

- For mounting, execute the following command in the dos shell:

  ```
  sh MountAll.sh <ROOTDIR>
  ```

  `ROOTDIR` may be an existent or non existent directory which may be specified relatively or absolutely.

> **Note**
>
> The `$SNIFF_DIR` environment variable must be set before running this script. The reason for this is that `SNiFF_DIR/bin` is mounted to `/bin` and in the script the Shell tools are called from `SNiFF_DIR/bin`.

- For unmounting, execute the following command in the dos shell:

  ```
  sh UmountAll.sh -k <directories you don’t want to unmount>
  ```

  If no directories are specified after `-k`, all directories are unmounted.

Windows 2000

Minor problems are known in the Directory navigation dialogs, these should not, however, interfere with normal work.

Online Help on Unix

You may encounter problems using SNiFF+ online help on Unix due to the Java Applet used. To use the Java-free version of the online documentation, do the following:

- In `$SNIFF_DIR/help/online/`

  replace the file “`topic.helpmap`” with the file (same name) in

  `$SNIFF_DIR/help/online/online_alt/`.

This allows the SNiFF+ menu to access the Java-free version of the online documentation.

To open the online documentation outside of SNiFF+, do the following:

- In `$SNIFF_DIR/doc/online/`

double-click on the file called “`index.htm`”.
This can be used to open the online documentation without using Java or JavaScript.

Screenshots in SNiFF+ for Unix online and paper documentation

The screenshots found in the SNiFF+ Documentation set were taken on Windows. Note that there are slight differences in the look and feel of the various views, dialogs and windows between the Unix and Windows versions of SNiFF+.

CMVC adaptors

- The usage of HP RCS can damage the repository files created with GNU RCS. Please be careful if you have installed both version control systems.

Version and configuration management

- SCCS does not support version labels. To enable this functionality, the label information is stored in the description text and is normally hidden from the user. To see the label information in the description text, you will have to issue SCCS commands from outside of SNiFF+.

Parser/Cross Referencer

- SNiFF+ 3.2.x uses `define` or `undefine` directives found in the Parser Configuration file as parameters for the preprocessor (when it is switched on).
- The option **Open XRef-DB Read Only** does not work correctly in a Private Working Environment (PWE) which does not access a Shared Source Working Environment (SSWE). If you need this functionality, we suggest you create an SSWE for your PWE to access.
- A reference to class members by friend functions is not shown in the call graph.
- In some cases, the Cross Referencer loses some references when browsing code that uses templates extensively.
- Only basic support for browsing virtual member functions in the call graph is available. Any reference to a virtual member function may result in several possible function calls at run time. The call graph simply shows the virtual member function of the declared class, or its first base class which implements the function.

Printing on Unix

By default, SNiFF+ is configured to print documents in DIN-A4 format. If your printer doesn’t support this format you have to change the setting in the print dialog.

IBM RS/6000 and DEC Alpha: ptty (Shell, Debugger) problems after abnormal exit

In certain situations, pseudo ttys are not properly closed during an abnormal termination of SNiFF+ (e.g., with the `KILL` or `SEGV` signals). This results in an incorrect handling of input and output in the Shell and Debugger tools.
Workaround:
Determine what shells are running for the current user:

```
ps | grep sh
```

and kill the shells that are left over after a terminated `sniff` process:

```
kill -HUP <shell_process_ids>
```

`<shell_process_ids>` are the process IDs of the SNiFF+ shells that are still running.

All breakpoints disappear from the Source Editor

If the gdb option 'verbose' is set and the program stops at the first breakpoint, SNiFF+ resets all breakpoints. The markers disappear from the Source Editor.
Workaround: Do not use verbose mode in `gdb`.

Error message when starting sniff on HP-UX

- The following error messages occur when starting SNiFF+ or a parser on HP-UX:

  ```
  ? mts: discarding 4 unaligned sbrk bytes
  ```

  This message can be ignored.

Compiling on Linux

To compile on Linux, you must enter your Make Command by doing the following:

- In the **Preferences > Platform view**, select *Linux-glibc* in the Platform List and press the **Set Writable** button.
- Select the **Make Support** tab and enter `make` in the Make Command field.
- Press **OK** to apply the settings and to close the Preferences dialog.

Java Debugger

- Deleting a breakpoint while the application is running does not always work correctly, i.e., the debugger still stops. If this happens, close the application and the debugger.

  Workaround: Before deleting a breakpoint, always choose **Execution > Interrupt** from the Debugger's menu.
- The Java Debugger (JDK1.1.3) does not work on Unixware.

Upgrading from earlier versions

> **Note**
>
> If you take advantage of the SNiFF+ Make Support, SNiFF+ 3.2.x cannot be used together with SNiFF+ 2.4/ 3.0x. See chapter Upgrading SNiFF+ specific files in general (Unix and Windows) — page 16 for details.

MKS Toolkit

The MKS Toolkit is no longer packaged with, or needed by SNiFF+ 3.2.x. This means that the following environment variables, which were previously needed by MKS, can now be removed from your system:

`HOME` **(on Windows)**

`ROOT_DIR`

`TMP_DIR`

Upgrading to SNiFF+ 3.2.x (Windows only)

FlexLM License Server

SNiFF+ 3.1 and above come with a newer version of FlexLM, FlexLM-6.1. When upgrading from SNiFF+ 3.0 and earlier, do the following:

- If you've copied the file `%SNIFF_DIR%\FlexLM\FlexLM-5.12\FLEXLM.CPL` to your system directory in earlier SNiFF+ versions, make sure that you copy the file `%SNIFF_DIR_3.2.x%\FlexLM\FlexLM-6.1F\FLEXLM.CPL` to your system directory.

  By doing this, the newer version of FlexLM will appear in the Windows Control Panel.

For more information, please refer to the `FlexLM_Licensing.wri` file in the `%SNIFF_DIR%\FlexLM` directory.

Upgrading from SNiFF+ 3.0.x to 3.2.x

The process of upgrading from SNiFF+ 3.0/3.0a to 3.2.x is the same as when upgrading from SNiFF+ 2.3/2.4 to 3.2.x with a few exceptions. These exceptions are listed below:

On Unix:

- The following additional files/directories will be saved with an appended `.old` extension:

  ```
  Version
  $SNIFF_DIR/Preferences/
  ```

On Windows:

- The process is the same as when upgrading from SNiFF+ 2.3/2.4 to 3.2.x except that steps 7 and 8 can be safely ignored.
- The following additional files/directories will be saved with an appended `.old` extension:

  ```
  Version
  %SNIFF_DIR%\Preferences\
  ```

How SNiFF+ specific files can be upgraded from 3.0/3.0a (Unix and Windows)

The format of the following SNiFF+ specific files has not been changed.

- **Site-wide Preference settings / User-specific Preference settings / Platform settings**

  All modifications applied to SNiFF+ 3.0/3.0a, except modifications to VC adaptor commands, will be retained.

- **Working environments**

  The old working environments directory is used so all your working environment information is retained. The new working environment files are located in `$SNIFF_DIR/workingenvs.orig` directory.

> **Note**
>
> If you want to do the SNiFF+ 3.2.x C++ Tutorial, in the **Preferences > Working Environment Config. Directory** field select your `$SNIFF_DIR/workingenvs.orig` directory.

- **Project files**

  The existing SNiFF+ 3.2.x project files can be used without making any changes.
- **Project state files**

  SNiFF+ automatically creates a new state file with the extension `V0.32`. SNiFF+ 3.2.x uses a new data format to store symbol information. Therefore, when you load an existing project in 3.2.x for the first time all your files are reparsed automatically. To make it possible to use SNiFF+ 3.0.x at the same time as 3.2.x we have added the extension `V0.32` to files containing symbol information, e.g. `Test.shared.symtab` is now `Test.shared.symtab.V0.32`. If you upgrade to 3.2.x and do not use 3.0.x any more we recommend to delete the old symbol information manually as it is no longer needed.
- **Symbol information/ Cross Reference information**

  Projects won’t be reparsed.

> **Note**
>
> The SNiFF+ Make Support has been modified and is no longer compatible with the Project Makefiles of earlier versions thus you will need to update the Makefiles. For details, see Updating project Makefiles (excluding Java) — page 17.

Upgrading from SNiFF+ 2.3/2.4 to 3.2.x on Unix

SNiFF+ 2.3/2.4 can be upgraded to 3.2.x by using the `install.kit` shell script. The script automatically creates backup copies by adding the extension `.old` to files/directories that you might have changed except for the `workingenvs` directory. The program uses the old workingenvs directory so all your working environments information is retained.

To upgrade to SNiFF+ 3.2.x, please complete the following steps:

1. Run the `install.kit` script as described in the *Installation Guide* and enter the platform and path of your existing installation.

2. You will then be asked whether you want to replace your existing SNiFF+ 2.x installation with SNiFF+ 3.2.x. Confirm the replacement. The following files/directories in `$SNIFF_DIR` will be saved with an appended `.old` extension:

   ```
   SitePrefs.sniff
   license.dat
   config/
   integrations/
   lib/
   make_support/
   symbol_API/
   ws_support/
   ```

3. When you start SNiFF+ 3.2.x for the first time, the automatic upgrade will be performed by SNiFF+.

> **Note**
>
> When upgrading SNiFF+ from previous versions, we suggest that the user who initially installed SNiFF+ should perform the upgrade and start SNiFF+ for the first time to make sure that the upgrade works.

Upgrading from SNiFF+ 2.3/2.4 to 3.2.x on Windows

SNiFF+ 2.3/2.4 can be upgraded to 3.2.x by using the `setup.exe` program that comes with SNiFF+ 3.2.x. The program automatically creates backup copies by adding the extension `.old` to files/directories that you might have changed except for the `workingenvs` directory. The program uses the old workingenvs directory so all your working environments information is retained.

To upgrade to SNiFF+ 3.2.x, please complete the following steps:

1. Start the installation by executing `SETUP.EXE` in the `setup/` directory on your installation CD-ROM or temporary installation directory. As installation directory choose the directory where SNiFF+ 2.3/2.4 is installed.

2. You will then be asked whether you want to replace your existing SNiFF+ installation with SNiFF+ 3.2.x. Confirm the replacement. The following files/directories in `%SNIFF_DIR%` will be saved with an appended `.old` extension:

   ```
   license.dat
   config\
   integrations\
   lib\
   make_support\
   symbol_API\
   ws_support\
   ```

3. Next, you will be asked if you want to create a program group. We recommend reusing the existing program group for SNiFF+, as there are new entries in SNiFF+ 3.2.x that would otherwise not be installed.
4. Acknowledge the default settings for the SNiFF+ specific environment variables and the PATH variable. Otherwise the upgrade will not work properly.
5. Overwrite the extensions for `.proj` and `.shared` when setup asks you about them.
6. As the integration with Microsoft Developer Studio is newly implemented in 3.1 and above you need to answer the corresponding question with “Yes” if you want to take advantage of the new integration.
7. Manually rename the file `SitePrefs.sniff.v23` in `%SNIFF_DIR%\` to `SitePrefs.sniff.old.`
8. When you start SNiFF+ 3.2.x for the first time, the automatic upgrade will be performed by SNiFF+.

> **Note**
>
> If you prefer to uninstall SNiFF+ 2.3/2.4 before installing SNiFF+ 3.2.x you can upgrade your Site- and User-specific settings also. Backup the original files (SitePrefs.sniff.v23 and .UserPrefs.sniff.v23) before uninstalling 2.3/2.4 and copy them to the new `%SNIFF_DIR%` directory. Then rename `SitePrefs.sniff.v23` to `SitePrefs.sniff.old` as described in step 7 above.

Using SNiFF+ 2.4/3.0.x and SNiFF+ 3.2.x in parallel

Although, it is now possible to choose a different program group for installing SNiFF+ 3.2.x, there is no way to overcome the problem that the PATH environment variable will contain the path to SNiFF+ 3.2.x before the path to SNiFF+ 2.4/3.0.x. Therefore, the latest installed version will always be started. We recommend that you do the following to install SNiFF+ 3.2.x parallel to SNiFF+ 2.4/3.0.x:

- When installing 3.2.x, DO NOT let the installation program modify the environment settings for `SNIFF_DIR`, `SHELL` and `ENV` and also DO NOT modify the path environment variable with the new settings for SNiFF+ 3.2.x.
- The installation program will then save the necessary settings to two batch files (`SetEnvVars.bat` and `SetPath.bat`) in the installation root directory.
- You can then start SNiFF+ 3.2.x by opening a shell, changing to the installation root directory and starting the two batch files. This will set the environment correctly in this shell and you can then start SNiFF+ 3.2.x by simply executing `sniff.exe`.

Upgrading SNiFF+ specific files in general (Unix and Windows)

- When upgrading to SNiFF+ 3.2.x, make sure that there are no periods (.) in VCS adaptor names.

  When using the RCS 5.6.7 adaptor supplied with SNiFF+, we suggest the following workaround:

  - Upgrade to SNiFF+ 3.2.x and open your project.
  - In the Project Editor, select all projects in the Project Tree. Note that these projects must be writable.
  - Select **Project > Attributes of Checkmarked Projects...**
  - In the Version Control System view of the Attributes of Checkmarked Projects dialog, select **RCS** as the VCS tool.

  If you are using your own VCS adaptor, and it has period characters (.) in its name:

  - Before upgrading, change the name of the VCS adaptor to one without periods (.). You could use underscores instead.
  - In SNiFF+, open your project and select all projects in the Project Tree of the Project Editor.
  - Select **Project > Attributes of Checkmarked Projects...**
  - In the Version Control System view of the Attributes of Checkmarked Projects... dialog, select the renamed VCS adaptor as the VCS tool.
  - Close the Project and upgrade to SNiFF+ 3.2.x.

Note

With new platforms and file types, avoid using names with periods.

Updating project Makefiles (excluding Java)

The SNiFF+ Make Support has been modified and is no longer compatible with the Project Makefiles of earlier versions so you will need to update the Makefiles. To do so:

> **Note**
>
> For upgrading Java projects, see page 20.

In the SiteMenus.sniff file

The `SiteMenus.sniff` file is in the `SNIFF_DIR/config/` directory.

- Open `SiteMenus.sniff` in an editor.
- Under the `# Patch Makefiles for New MakeSupport` line, uncomment the following lines:

```
# >Makefiles
# shell "Update Makefile(s) for New Makesupport" "echo Updating File %f; sh $SNIFF_DIR/make_support/UpdateMakefile.sh %f"
```

or

In the UserMenus.sniff file

The `UserMenus.sniff` file is in the `%SNIFF_DIR%\Profiles\<`*Username*`>\` directory on Windows, and in your `$HOME/.sniffrc/` directory on Unix.

- Open `UserMenus.sniff` in an editor.
- Copy the following lines from `SiteMenus.sniff` to `UserMenus.sniff`:

```
#Patch Makefiles for New MakeSupport
#>Makefiles
# shell "Update Makefile(s) for New Makesupport" "echo Updating File %f; sh $SNIFF_DIR/make_support/UpdateMakefile.sh %f"
```

- In `UserMenus.sniff`, under the `# Patch Makefiles for New MakeSupport` line, uncomment the following lines:

```
# >Makefiles
# shell "Update Makefile(s) for New Makesupport" "echo Updating File %f; sh $SNIFF_DIR/make_support/UpdateMakefile.sh %f"
```

In the Project Editor

1. Load the project created with an earlier SNiFF+ version.
2. In the Filter dialog, **File Types** view, make sure that Makefiles are selected.
3. In the File List, select all Makefiles.
4. If the Makefiles are read-only, check them out by choosing **File > Check Out**.
5. Choose **Makefiles > Update Makefiles for New Makesupport**.

   This command runs a script which removes the following lines from the selected Makefiles:

   ```
   include $(SNIFF_MAKEDIR)/$(SNIFF_VPATH_INCL)
   include $(SNIFF_MAKEDIR)/$(SNIFF_OFILES_INCL)
   ```

   It is necessary to remove these lines since the `vpath.incl` and `ofiles.incl` files are no longer generated by the new Make Support, so trying to include them would result in an error. The `VPATH` macro is not used in the new make support because of its limitations:

   Limitations of `VPATH`:

   - not all compilers support the `inhibit local includes` flag
   - inconsistencies because dependencies may be incorrectly resolved
   - builds take longer

   Advantages of `VPATH`:

   - `VPATH` can search multiple directories for dependencies and source files

Using Native Make (Unix only)

The following information is only relevant for you if you want to use SNiFF+'s Make support system together with native Make on the following platforms:

AIX 4.2 or newer
HP-UX 10.20 or newer
Irix 5.3 or newer
Sinix 5.42 or newer

Some features used in the standard set of Make Support files do not work correctly with native Make on these platforms. We therefore supply a separate set of Make Support files in this directory that must be used together with native Make.

Please note that the following will only work with the platforms mentioned above on condition that your source files are located in the same directory where *make* is called. Thus make sure that the **Directory** field is left blank for files of type implementation. You can find this field in SNiFF+'s Preferences and Project Attributes **> File Types** view **> General** tab.

To apply these native Make Support files, do the following:

- In your `SNiFF_DIR/make_support` directory, rename your standard `.mk` file e.g., `general.c.mk to general_orig.c.mk.`
- Then rename your native `.mk` file in the same directory to your standard `.mk` file e.g., `general.c.native.mk to general.c.mk.`

Note to users of native Make on DEC OSF1:

SNiFF+ does not currently support native Make on **DEC OSF1**. We recommend using the GNU Make supplied with SNiFF+ instead.

Note to users of native Make on SCO Unixware:

SNiFF+ does not currently support native Make on SCO Unixware. Native Make does not work with both standard and native Make Support files since the VPATH macro is not expanded correctly.

Note to users of native Make on RS6000 and HP-UX:

- In your `SNiFF_DIR/make_support` directory, rename the `general.link.native.mk` file to `general.link.native.orig.mk.`
- Then rename the `general.link.native.mk` file in the same directory to `general.link.mk.`

Upgrading Java Projects

- To use your old SNiFF+ projects with SNiFF+ 3.2.x, copy `<sniff_installation_directory>/config/template.java.Makefile` to your project directory and replace your old Makefile with this one. Then update your Makefiles.

Note

If you have made changes in your old Makefiles, make these changes manually in the new Makefiles.

- If you are using SNiFF+ Make Support, each directory should contain only Java files. It is not possible to build a project correctly if source files in other programming languages are located in the same directory. To avoid this problem, store these files in separate directories and create separate projects for them. Also make sure that your non Java projects do not contain the Java file type.
- For Java, the working environment concept with Private(PWE) and Shared Object Working Environments is no longer supported in SNiFF+ 3.2.x. This is because equivalent packages that are entered separately in the class path (as is the case in layered Working Environments) are not explicitly examined by the JDK tools for each file. The first entry that is matched is cached and reused by the compiler and interpreter. This may lead to inconsistent results if working environments are layered. If you have been using shared environments, you have two options:
    - either move your PWE so that it directly accesses the Repository or Shared Source Working Environment (SSWE). If you are working with more than one PWE, do the same for all. Check out all project files to your PWE.
    - or create new PWEs which access the Repository or SSWE and create new projects in your PWEs.

Upgrading SNiFF+ specific files from 2.3/2.4 (Unix and Windows)

- Site-wide Preference settings

  When SNiFF+ is installed in the same directory where 2.3/2.4 is located a copy named `SitePrefs.sniff.old` is created. When starting SNiFF 3.2.x for the first time all entries are merged automatically with one exception. The VCS adaptors that came with 2.3/2.4 are not merged if they are available with the same name in 3.2.x. But if you have created new adaptors using other than the default names they will be merged automatically. Therefore, if you have applied changes to the default adaptors that you want to upgrade please rename them in 2.3/2.4 before you upgrade. They will then appear in 3.2.x with the new name and you can restore the old name then in 3.2.x.

- User specific Preference settings

  For each user all the existing User specific settings are converted automatically to the new SNiFF+ 3.2.x format and are written to the new location (`$HOME/.sniffrc/UserPrefs.sniff` on Unix, `%SNIFF_DIR%\Profiles\<user_name>\UserPrefs.sniff` on Windows).

- Working Environments

  SNiFF+ 3.2.x uses a new data format and new file names to store Working Environments information. The old working environments directory is used so all your working environment information is retained. The new working environment files are located in `$SNIFF_DIR/workingenvs.orig` directory.

> **Note**
>
> If you want to do the SNiFF+ 3.2.x C++ Tutorial, in the **Preferences > Working Environment Config. Directory** field select your `$SNIFF_DIR/workingenvs.orig` directory.

- Project files

  The format of the Project files has changed between SNiFF+ 2.3/2.4 and 3.2.x. It is, of course, possible to read the old format with 3.2.x but 2.3/2.4 is not able to read the new format. Therefore, if you want to work on a project with 2.3/2.4 and 3.2.x at the same time be careful to make changes to the project attributes (project settings, list of files/ subprojects) with 2.3/2.4 only.
  The make command can now also be specified in the new platform settings. This allows you to compile from within SNiFF+ on platforms with different make commands without

changing the Project file. Only if you want to take advantage of this new feature, you have to clear the make command in the Project file. Otherwise, the setting in the Project file will overwrite the setting in the Platform definition.

When using more than one Parser Config File, the separator has been changed from ":" to ";" in 3.2.x. You have to modify your project files manually if you use several Parser Config files.

The settings for the Java Source Root and Generate Root Directory are no longer specific to a Working Environment but are stored per Project and can be set on the Java tab view on the Build Structure page of the Project Attributes. If you have used these settings in 2.3/2.4 you need to update your project files accordingly. The settings in the Working Environments are no longer used by SNiFF+ 3.2.x.

- Project state files

  SNiFF+ 3.2.x uses a new data format to store Project state information. To make it possible to use SNiFF+ 2.3/2.4 at the same time as 3.2.x we have added the extension `V0.32` to files containing project state information, e.g.

  `Test.shared.state`

  is now

  `Test.shared.state.V0.32`

  If you upgrade to 3.2.x and do not use 2.3/2.4 any more we recommend that you delete the old state files manually as they are no longer needed.

- Symbol information

  SNiFF+ 3.2.x uses a new data format to store symbol information. Therefore, when you load an existing project in 3.2.x for the first time all your files are reparsed automatically. To make it possible to use SNiFF+ 2.3/2.4 at the same time as 3.2.x we have added the extension `V0.32` to files containing symbol information, e.g. `Test.shared.symtab` is now `Test.shared.symtab.V0.32`. If you upgrade to 3.2.x and do not use 2.3/ 2.4 any more we recommend to delete the old symbol information manually as it is no longer needed.

- Cross Reference information

  SNiFF+ 3.2.x uses a new data format to store Cross Reference information for the RAM-based solution. Therefore, when you start the Cross Referencer for an existing project for the first time new Cross Reference information is generated automatically. To make it possible to use SNiFF+ 2.3/2.4 at the same time as 3.2.x we have added the extension `V0.32` to files containing Cross Reference information, e.g. `Object.h.ref` is now `Object.h.ref.V0.32`. If you upgrade to 3.2.x and do not use 2.3/2.4 any more we recommend that you delete the old Cross Reference information manually as it is no longer needed.

Note

You may delete the backup files and directories having the `.old` extension after a successful upgrade.

- Makefiles

  Due to the complexity of Makefiles it is not possible to upgrade them automatically. If you have applied changes to the Project- or Platform Makefiles that came with 2.3/2.4 you have to merge them manually into 3.2.x. As there are no major changes to the structure and functionality this should be straightforward.

- Custom menu files

  Site specific Custom menu files can be upgraded by copying them manually. SiteMenus.sniff must be copied from `/config.old` to `/config`. User specific settings are reused automatically without any change.

- File dialog history

  Also the files containing the file dialog history entries can be upgraded by copying them manually.
  On Windows copy the file

  `%HOME%\.FileDialogHistory`

  to `%SNIFF_DIR%\Profiles\username\FileDialogHistory.`

  On Unix copy the file

  `$HOME/.FileDialogHistory`

  to `$HOME/.sniffrc/FileDialogHistory.`

Other upgrading issues from 2.3/2.4

- Deleting obsolete files

  The **Make > Check_Obsolete_ Files** menu entry in the Target menu has been removed. You can now only delete obsolete files by selecting **Project > Check Obsolete Files...** in the Project Editor. In the Obsolete Files dialog that appears, select the files which you want to delete and press the **Delete** button. If you want to delete obsolete files using a script, use the following Sniffaccess request:

  check_obsolete_files `PROJECT`

  This request returns a list of obsolete files which you must manually remove.

  For information on how to invoke Sniffaccess, please refer to the Sniffaccess chapter in the Reference Guide.

- Debugger settings

  In SNiFF+ 3.2.x the debugger settings are stored in the new Platform definitions. On Unix, the default platform setting is called UNIX and default debugger is GNU gdb. On Windows, the default platform setting for Windows uses sniffjdb, the SNiFF+ Java debugger as default.

  If you want to use other debugger settings, either modify the default platform settings in the Preferences, or assign other, more appropriate, platform settings to your various working environments in the Working Environments tool.

- Documentation Template Files

  To better distinguish Documentation Template Files from the new Project Template Files, the extension has been changed to `*.dtmpl`. Simply rename your existing files and copy them into the `$SNIFF_DIR/config/docu` directory if you want to keep them.

- Sniffaccess requests

  The Working Environment requests **set_workspaces** and **update_private_ws** have been changed to **set_workingenv** and **update_private_we** respectively.

Application papers

The following application papers are supplied with your SNiFF+ installation. These (in compressed PostScript or PDF format) are in your `$SNIFF_DIR/doc` directory or online:

- **The Symbol Table API paper (symtabapi.pdf & symtabapi.ps.Z)**

  This paper describes how to use the SNiFF+ Symbol Table API.

- **The SNiFF+ Parser Development Kit paper (parserdevkit.pdf & parserdevkit.ps.Z)**

  This paper describes how to use the SNiFF+ Parser Development Kit.

- **The Ada Tutorial (ada.pdf & ada.ps.Z)**

  This paper describes how to use SNiFF+ with Ada.

- **Integration for PVCS on Windows and Unix (pvcs.pdf & pvcs.ps.Z)**

  Describes how to use the SNiFF+ integration with the PVCS version and configuration management tool.

- **Integrating SNiFF+ with CVS (cvs.pdf & cvs.ps.Z)**

  This application paper describes how to integrate SNiFF+ with CVS. CVS is a public domain version control system.

- **Integrating SNiFF+ with ILOG (ilog.pdf & ilog.ps.Z)**

  This application paper describes how to integrate SNiFF+ with the ILOG tool suite from ILOG: ILOG Views, ILOG Broker, ILOG server and ILOG DB Link.

- **Integrating SNiFF+ with Rational Rose (roseunix.pdf and rosewin.pdf & roseunix.ps.Z and rosewin.ps.Z)**

  This application paper describes how to integrate SNiFF+ with the modeling tool Rational Rose.

You can find the application paper describing how to integrate SNiFF+ with the external editor Multi Edit in the following directory:

- `$SNIFF_DIR/integrations/MultiEdit`

Version control and configuration management tool adaptors

This SNiFF+ release contains adaptors for the following CMVC tools:

CMVC tool	**Name of adaptor**	**Platforms**
Rational Software ClearCase 2.0 or newer	ClearCase (does not support SNiFF+ Configuration Manager)	all platforms
CVS 1.9 or newer	CVS	all platforms
GNU RCS 5.6 or newer	RCS	all platforms
GNU RCS 5.7 or newer	RCS_CROSS	for cross-platform use
HP RCS for HP/UX	RCS_HPUX	HP/UX
PVCS 6.x	PVCS	all platforms
Continuus 4.5	Continuus	all platforms
SCCS	SCCS	all Unix platforms except for IBM AIX and HP/UX 9.x
SCCS for AIX 4.2	SCCS_AIX	IBM AIX only
SCCS for HP/UX 9.x	SCCS_HPUX	HP/UX 9.x only
SNiFF+ file locking	SNIFF	all Unix_platforms
TeamConnection 3.0	TeamConnection	all platforms

Supported debuggers

This SNiFF+ release supports the following debugging systems:

Platform	Debugger & Version
AIX 4.2 or newer	gdb 3.x gdb 4.x dbx 3.1
Alpha-DEC 3.2 or newer	gdb 3.x gdb 4.x dbx 3.11.8 decladebug 4.0-7
HP/UX 10.x or newer	gdb 3.x gdb 4.x dde 3.25A.P1 xdb A.09.01
IRIX 5.3 or newer	gdb 3.x gdb 4.x dbx 3.19
Linux 2.x	gdb 3.x gdb 4.x DDD 3.x
Sinix 5.43 (Reliant Unix) or newer	gdb 3.x gdb 4.x dbx 2.0C
Solaris 2.4 or newer	gdb 3.x gdb 4.x dbx
Unixware 7	gdb 3.x gdb 4.x
All platforms and Windows NT/95/98	Java debugging adaptor
SunOS 4.1	gdb 3.x gdb 4.x dbx

Resource requirements for SNiFF+ 3.2.x

The following resource requirements are from tests on a Solaris workstation and a Windows machine.

Required disk space

Installed SNiFF+	Diskspace on Solaris	Diskspace on Windows NT
Complete Installation	~100 MB	~71 MB
postscript/pdf files	~12 MB	~12 MB
Example sources for all tutorials	~6 MB	~6 MB
Core product executables (without compiler)	~82 MB	~53 MB
Platform specific binaries	30 - 50 MB	-----------

Memory required for a project with approx. 85 KLOC with Rapid Reference Technology enabled

States of the running SNiFF+	Memory on Solaris	Memory on Windows NT
SNiFF+ empty process	~25 MB	~15 MB
After loading project Symbols	~31 MB	~19 MB
After one cross reference (referred-by query)	~32 MB	~20 MB

Diskspace for files generated by SNiFF+

The diskspace required for files generated by SNiFF+ depends on the number of symbols in your source code but is usually not larger than the diskspace required for the sources.

List of Bug-fixes

The following bugs have been fixed in SNiFF+ 3.2.1.

Number	Description
TKFat10141	The Statistic dialog in the Project Editor incorrectly displays the number of symbols
TKFat10459	Environment variables in the Platform Include path are not correctly dumped to the `$PLATFORM.incl` file
TKFat10455	ClearCase Trigger mechanism does not invoke project reload when a project description file is checked in
TKFat10284	The browser does not open when you double-click on an html file
TKFat10374	The mouse cursor changes to (and remains) a sandglass in the Open File dialog
TKFat10381	OrbixWeb compiler name is `idlj` instead if `idl`
TKFat10382	Pre - target rules are not taken into account because they are defined in the wrong position in the Project Makefile
TKFat10421	Incorrect Include Directive and dependency generation when a header file is added with a filetype directory
TKFat10436	The make rule in the target dialog does not work with target redirection
TKFat10387	The menu command **ClearCase > Merge Version** does not work correctly
TKFat10176	The preprocessor does not correctly expand environment variables in include directives
TKFat10257	Scrolling in the History Pane with the Wheel mouse opens the Diff/ Merge tool
TKFat10385	The command **Project > Add/Remove Files to/from a *project*** does not work properly
TKFat10432	On Windows, the redirection of project description files does not work when the redirection directory is outside the Working Environment root directory.
TKFat10415	Search with and without Index in the Retriever leads to different results

Number	Description
TKFat10262	SNiFF+ crashes when the Shell is opened and the platform for the Working Environment is not available
TKFat10336	Include information in the Include Browser is incorrect when includes use a path and source files are redirected
TKFat10367	The dialog **Cannot Connect to Parser** appears for each file that needs to be parsed
TKFat10430	The **Project > Synchronize Checkmarked Projects** and **Project > Synchronize File Status** commands do not work with RCS_HPUX
TKFat10329	SNiFF+ crashes when the cross reference database is read-only

The following bugs have been fixed in SNiFF+ 3.2:

Number	Description
TKFat08026	No templates supplied for project setup
TKFat09659	SNiFF+ crashes when trying to create a Retriever index file in a directory where the user has no write permission
TKFat10111	`make all` for java projects does not stop in case of an error
TKFat08734	The **Show Error** command of the Shell tool does not work
TKFat09526	The scroll function of a mouse is not supported
TKFat09670	Trying to retrieve from files of a removed subproject causes SNiFF+ to crash
TKFat09800	Takefive daemon crashes on DEC-UNIX
TKFat09875	The *target*.html file is not correctly generated for JDK 1.2.x
TKFat09883	Native *make* does not work
TKFat09884	Installation of SNiFF+ 3.1 fails on Irix 6.2
TKFat09887	SNiFF+ crashes when using the Retriever on Linux.
TKFat09892	sniffjmake crashes when the classpath is too long
TKFat09901	SNiFF crashes when scrolling
TKFat09908	**Next** button of **Create New SNiFF+ Project** page in the Wizard is disabled

Number	Description
TKFat09925	An incorrect error message stating that the package root is not set correctly appears
TKFat07616	`<CTRL> R` is the keyboard shortcut for Next Match and Referred By
TKFat08337	Inconsistent naming of **Filters...** button in all tools except Retriever (**Filter...** button)
TKFat09342	No warning appears when you try to create a working environment which already exists
TKFat09609	In the Configuration Manager, when you select **Merge Branch Configuration > 3-Way Merge**, a check out dialog appears. Even when you select **No** in this dialog, files are checked out
TKFat09612	Problem in reloading a project in another working environment
TKFat09617	Problem with dbx on SGI when several projects are loaded
TKFat09640	Context Sensitive Help (`F1`) does not always work for Preferences and Project Attributes
TKFat09653	When installing SNiFF+ 3.1 over an existing SNiFF+ 3.0.2 installation on Windows, old HTML help-files are not removed
TKFat09654	Cannot execute the **Target > Make File** command in a cross platform environment
TKFat09665	SNiFF+ crashes when you execute the **Add New Subproject** command
TKFat09672	No button to force update of Retriever Index
TKFat09679	sniffparser does not start on SCO Openserver 5.0.x
TKFat09702	We ask for the wrong file in sniff_feedback
TKFat09723	The Configuration Manager crashes with SCCS when updating symbols
TKFat09790	In SNiFF+ 3.1, you cannot specify the version number when checking in a file. SNiFF+ always checks the file in as HEAD
TKFat09803	Retriever stops working if Working Environment root is not writable
TKFat09810	IDL Make Support does not work
TKFat09828	For Java Cross Platform projects, an error message “sniffjmake not found” is reported when trying to build the target

Number	Description
TKFat09830	On Digital Unix, SNiFF+ crashes when you drag and drop a Working Environment in the Working Environments tool
TKFat09831	SNiFF+ crashes when parsing certain code

The following bugs have been fixed in SNiFF+ 3.1:

Number	Description
TKFat08704	SNiFF+ crashes when a line is selected in the Source Editor
TKFat00919	Conversion of documentation files to html files in the Documentation Editor does not correctly generate a book file
TKFat07403	In the Xref Database used in Crisp Parsing, references to functions with an argument called "private" are not shown
TKFat08172	Inconsistent jumping to/from the implementation/definition of symbols
TKFat08553	The Retriever changes only the first occurrence of a string although the **Change All** option has been selected
TKFat08722	Symbolic links to sources should be made on target host
TKFat09035	SNiFF+ does not accept blanks in user names
TKFat09103	Incorrect operating system returned for Windows 98
TKFat09163	Double-clicking a project in the Launch Pad displays the History window of the Project Editor.
TKFat09396	The **Check Obsolete Files** command also shows CVS directories as obsolete
TKFat09284	If using ClearCase and you rename a file from inside SNiFF+'s Add/ Remove Dialog, the called command ignores the ClearCase archive name
TKFat07329	File that has been partially saved due to a lack of disk space is checked in
TKFat08204	Various messages appear when starting the dbx (Sparkworks) debugger on Solaris
TKFat08293	A double-click on a list item in the Preferences opens an empty dialog
TKFat08294	Group Project Attributes dialog is too large

Number	Description
TKFat08351	A log file is used when a file is specified in the **Log File** field in the Tools view of the Preferences, irrespective of whether the **Use Log File** checkbox is selected or not
TKFat08387	The dbx (Sparkworks) debugger adaptor does not work correctly
TKFat08404	Cannot correctly create a branch from a version older than HEAD
TKFat08408	SNiFF+ crashes when using the Retriever under certain circumstances
TKFat08429	File/Directory permissions of the archives are lost
TKFat08433	The -h option in the `rcsdiff` command is still present in the `RCS.py` file
TKFat08435	`inode` comparison does not work correctly for symbolic links in the `sniffUpdateFile.RCS.sh` file
TKFat08499	RCS Check In fails under certain circumstances
TKFat08506	Incorrect generation of dependencies
TKFat08532	Incorrect branch number for RCS archive files
TKFat07905	IDL Parser does not work correctly
TKFat07942	Pico example: wrong Makefiles. Platform Makefiles are only included for 2 platforms
TKFat07972	Wrong highlighting for IDL files with #pragma
TKFat08316	C Tutorial: Single User tutorial does not compile
TKFat08374	Fortran parser doesn’t understand include statements like INCLUDE (filename)
TKFat08411	The Ladebug debugger doesn’t work properly in the SNiFF+ Shell
TKFat08497	sniff start: Disabling the Project Wizard leads to a frozen sniff
TKFat08539	Java Parser does not work correctly
TKFat08540	Fortran Parser doesn’t recognize “entries”
TKFat08575	Loose files are not shown if a file with the same name is in the project
TKFat07383	Crisp Java Parser: Static initializes of a class are not instance variables

Number	Description
TKFat08180	The result of a Find is sometimes not highlighted

Bundled GNU components

This SNiFF+ releases for Unix contain the binaries of GNU **gcc 2.7.2.2**, **gdb 4.17, libg++ 2.7.2**, **make 3.74** and **rcs 5.7.** SNiFF for Windows contains **rcs 5.7** for Windows 95/NT.

> **Note**
>
> The SNiFF+ release for Linux **only** contains the binaries of GNU **gdb 4.17** and **make 3.74**!

The SNiFF+ Installation Guide describes how to install the GNU components.

Important: All GNU components are distributed under the GNU GENERAL PUBLIC LICENSE (see file `copy.txt` and `copylib.txt` in the root directory of the SNiFF+ distribution). GNU components are provided on an as-is basis and are not supported in any form by TakeFive Software.

SNiFF+ is preconfigured for using `gdb`. Depending on your needs, you may want to install other compilers and debuggers. In the case of a debugger change, please do not forget to tell SNiFF+ which version (executable) it should use as a debugger back-end (For details, please refer to the Reference Guide).

Important information about ET++

ET++3.0 is part of the SNiFF+ distribution, however, it is not supported by TakeFive. This software is in the public domain and is provided on an as-is basis. Please read the copyright files in the corresponding directories.

TakeFive SCE Newsletter

"TAKEFIVE SCE Newsletter", a quarterly on-line newsletter service designed to communicate new product information and news from TakeFive Software, is now available from TakeFive's home page.

You can download the latest and previous versions of the newsletter from

http://www.takefive.com/news/sce-news.html

Or, you can obtain our newsletter every quarter by subscribing to the mailing list:

```
To:
```

majordomo@takefive.co.at

```
From: <YOUR EMAIL ADDRESS>
Subject: subscribe takefive-newsletter
```

Useful links

Here are some links to useful SNiFF+ web pages:

- The most up-to-date version of this file

http://www.takefive.com/products/documents.html

- TakeFive Support Knowledge Base

http://www.takefive.com/support/kb.html

- Frequently Asked Questions

http://www.takefive.com/faq

- SNiFF+ Users Mailing List

http://www.takefive.com/support/sniff-list.html

- SNiFF+ Users Mailing List Archive

http://www.takefive.com/sniff-list