# Release Notes for ALINT 2012.01

These Release Notes list additions, changes, and fixes to ALINT that have been implemented since the previous version, ALINT 2010.10.SR1.

## What's New

### 64-bit ALINT for Windows

- The 64-bit edition of ALINT for Windows is available. The installation package is delivered in a separate file (*ALINT-<version>-x64.exe*) and by default the new version is installed to the *C:\Aldec\ALINT-<version>-x64* directory. The 64-bit version offers the same features and functionality available in the 32-bit version, including a fully functional GUI.

  The 64-bit edition can be installed alongside a 32-bit version of ALINT. The version of the program (32- or 64-bit) is included in the name of the ALINT icon available both in the Start menu and on the desktop.

  The ALINT for 64-bit Windows was verified on the following operating systems:

  - Microsoft Windows Vista™
  - Microsoft Windows 7™
  - Microsoft Windows Server® 2003.

### Exclusion Management

- This release extends the default exclusion management mechanism with a possibility to justify the reasons for every irrelevant violation detected. This functionality facilitates a general good practice of creating quality design artifacts, and is essential for safety-critical designs with strict documentation requirements that do not allow creating exceptions for any coding standard without having a valid reason.

  Excluded violations together with the corresponding comments are saved to the violation database, enabling the relevance-based view for every particular design:

  - Relevant violations - problems that require attention (need to be addressed).
  - Excluded violations - items intentionally skipped by the reviewer (added to an exclusions file). The excluded violations neither count toward the total number of violations detected during a linting session, nor appear in design quality reports.

  Note, that the violations can be filtered by the relevance criteria.

  Since every violation database can be associated with an exclusions file, the contents of the currently associated ( main ) exclusions file can be managed from the Violation Viewer, even without having a design loaded.

## Standalone Reports

- The all new, completely redesigned HTML report is now available. The new report provides an insight into design quality and includes tools for design hierarchy navigation, violations analysis, and cross-probing from violation messages to source files. The report can be viewed using any common web browser, with no additional ALINT license required.
- The text-based reporting engine was redesigned to enable exporting violations to detailed and well-structured text reports using the `avdbreport` macro command. The default report contains header, optional summary, and violation sections tuned for script based post-analysis.

  For more details refer to the *Working with Results* | *AVDB Reporting* section of the *User's Guide*.

## Graphical User Interface

### Violation Viewer

- The currently associated exclusions file is now visible in the **Main Exclusions File** field of the Violation Viewer toolbar, and referred to as "main". The main exclusions file contents are modified based on the **Disable/Enable Rules for** action in the Violation Viewer.

  Note, that if there is no exclusions file associated with the currently analyzed violation database, **Disable/Enable Rules for** opens the **Associate with Exclusion File** dialog box. The dialog enables associating the database with a new or existing exclusions file.
- The relevance of a violation now defines its appearance on the main panel of Violation Viewer. By default, the relevant violations are displayed using a base font and the excluded ones are grayed out. Group nodes (design units, instances, rules) depend on their contents: the base font is used when a node contains both relevant and excluded violations. Additionally, it is now possible to filter violations by relevance. Note, that the appearance settings can be configured in the **Violation Viewer | Appearance** category of the **Preferences** dialog box.
- The Fixed and Irrelevant violation statuses are deprecated so that **New** or **Read** are the only statuses available. This clarifies the difference between the status and relevance, and resolves confusions that had been placed in the past.
- The whole design hierarchy (design units, instances, and source files) is now stored in the resulting violation database regardless of violations detected or not. (With previous versions of the tool, the violation database used to store only those parts of the design hierarchy for which violations were detected).

  Note, that all global scope violations are stored together in VHDL Global Scope, or Verilog Global Scope node depending on the language. Violations belong to global scope when they are not associated with any of the design units (e.g. violations for comments style, preprocessor directives etc.).

### Exclusions Editor

- Every exclusion or group of exclusions can be associated with the appropriate justification - comment explaining why a certain rule was not relevant for a certain context or design. This feature facilitates proper design documentation and reporting.

**Framework**

- The functionality of the **Favorites** toolbar has been enhanced. The toolbar can be customized in the **Organize Favorites** dialog box where you can add, rearrange, or remove individual toolbar buttons, customize the appearance of the buttons and specify their tooltips.

  Additionally, the toolbar now allows embedding and executing macro commands.
- The mouse wheel control has been improved and made consistent across the entire environment: the multi-line scrolling feature has been implemented in the Console window, the page-by-page scrolling by using the mouse wheel and the Ctrl key has been implemented in multiple tools, and the horizontal scrolling by tilting the mouse wheel or by using the horizontal scroll wheel is supported.

**Design Manager**

- The algorithm for resolving compilation dependencies between design units coming from different languages has been improved. If a design contains source files that form a multi-layer dependency chain, for example: Verilog-VHDL-Verilog-VHDL, such complex dependencies are properly resolved and an appropriate compilation order is applied. Note that before using the **Compilation-time linting: Design(s)** context menu option, the dependencies must be properly defined in the **Dependencies** window. For more information, refer to the *Framework Basics* | *Design and Workspaces* section of the ALINT *User's Guide*. (SPT61540)
- Now, the workspace is automatically saved before compilation- and elaboration-time linting of the design. All of the modifications made in the design are kept and there is no need to run an additional command when ALINT is started in command-line mode. (SPT60732)

**HDL Editor**

- A new tool facilitating source code editing has been implemented: the **Code Templates** window. The tool allows use of predefined code templates by dragging and dropping the code snippets from the **Code Templates** window to the HDL Editor or by selecting the templates from the AutoComplete list. The tool is equipped with the following features:

  - **Predefined Code Templates**

    The tool is delivered with a variety of predefined code constructs for most of the languages supported by ALINT: VHDL, Verilog, Perl and Do macro language.
  - **Integration with AutoComplete**

    The tool is integrated into the AutoComplete feature of the HDL Editor. Besides providing the autocomplete feature for the language keywords, the functionality prompts you with complete code snippets defined in the Code Templates.
  - **Code Template Editor**

    The templates can be freely customized and supplemented with user-defined constructs. The user can define special variables during the usage of a template in the HDL Editor that will be considered editable special fields in the template edition mode. If a given special field occurs more than once in the template definition, during its edition in the HDL Editor, all of its occurrences will be automatically updated with the same value.
- The **Enable virtual spaces** option in the **Preferences** dialog box has been disabled for all supported types of text files.

- Two new options were added to the file type pages in the **HDL Editor** category of the **Preferences** dialog box:

    - **Ensure newline at end of file**: This option is especially useful for languages whose compilers require the newline character at the end of the file, e.g. for the C source files.
    - **Remove trailing whitespace**: When enabled, it automatically removes unnecessary whitespace characters at the ends of lines. Depending upon the user preferences, the option can be applied to all lines in a file or to the currently edited lines only.
- The contents of the HDL Editor window can be scrolled horizontally by tilting the mouse wheel left or right.
- Syntax highlighting has been enhanced for dynamic languages such as Tcl (**.do* and **.tcl*) and Perl. For example, the editor now correctly highlights multi-line comments and time values:

```
# This is a multi-line \
  comment.
run 10ns
```

  Additionally, syntax highlighting for VHDL and Verilog is faster.
- Highlighting of words and user-defined phrases in now possible. When the **Enable phrases/words highlighting** option is enabled in the **General** category of the HDL Editor preferences, manual selection of a portion of a text, or placing a text cursor in a word, causes a background of other occurrences of this selection to be highlighted. The color of the highlighting can be customized in the **Appearance** category.
- The current location of the **Find and Replace** dialog box is remembered after you close the dialog box and then reopen it. Previously, the dialog box always appeared in the middle of the screen. Now, the frequent use of the Find or Replace options (as well as browsing through named bookmarks or jumping to a line of code) will no longer require the dialog box to be dragged to a new location where it does not overlap your work. (SPT49654)
- Code folding capabilities of the HDL Editor have been enhanced. With this release, the following features have been introduced:

    - HDL code is automatically analyzed at the moment of opening a file.
    - New code is analyzed on the fly, when being typed by the user.
    - Blocks of comments are recognized and treated as separate sections. (SPT13882)

  As the above features eliminate the need to launch the code analysis manually, the **Generate Structure** command has been removed from the HDL Editor Toolbar and the context menu.
- The **Find Matching** command can recognize the VHDL construct `protected ... end protected`. The matching members of the construct are also highlighted in the HDL Editor when the insertion point is positioned at one of them.
- The cursor position is now remembered. When you jump to a source point during the violations analysis, the editor saves the history of the opened source files and visited rows, and then allows browsing by recent cursor locations. To relocate to the previous or next cursor location, use either:

  the `hde.cursor.position.previous` and `hde.cursor.position.next` commands or

  the Ctrl + [ and Ctrl + ] shortcuts.

**File Browser**

- The **Find in Files** option has been added to the context menu of the File Browser. The new option allows searching for a specified string in predefined types of files, or you can customize the search settings by using the available filters or regular expressions. By default, the option starts searching for a string in files stored in the currently selected folder. The search results are displayed in the **Find Results** window.

### Flow Manager

- Previously, files generated after flow execution were stored in the flow's directory. Now these files are stored in the design directory. This allows sharing flow files across the network. (SPT61837)
- The **Options** button has been removed from the **Elaboration-time Linting** dialog window that appears when the linting is being executed from the Flow Manager. This change has been made in order to remain design properties unchanged during the flow execution. (SPT62158)

### Preferences

- The **Reload files without prompt** option has been added to the **Environment | General** page of the global **Preferences**. If this option is enabled, the confirmation dialog is not shown on file reload. (SPT62159)
- The **Join clock signals** option has been added to the **Linting | Entries** page of the design **Properties**. It allows specifying clock signals that belong to the same clock domain. (SPT62249)

### Console

- The **Find text** dialog box was introduced to allow searches of the Console window for a specified text string. The search pattern can be specified as a regular expression, which allows constructing more complex searches. The dialog box can be accessed by clicking the **Find** button on the Console toolbar. The **Find Next** and **Find Previous** buttons are also available to quickly jump to a next or to previous occurrence of a recently specified string. (SPT50804, SPT60341)
- The **Transcript Files** option was added to the **Find in Files** dialog box. With this option enabled, you can limit the searched scope to the contents of the Console transcript files. The Console transcript files are also included in other searches performed by this engine. (SPT50804, SPT60341)
- The **Wrap at origin/end** option has been added to the **Find text** dialog box in the **Console** window. Enabling the new option allows resuming the search from the beginning once the last row of the Console has been reached.

## System Libraries

- The `ieee_proposed` library has been updated to latest version published in September 2010.

## Tcl Interpreter

- The Tcl Library (a set of Tcl extensions written in Tcl) has been upgraded from version 1.11.1 to version 1.12.

## Macro Commands

- The new `avdbreport` command is available. It generates a detailed and well-structured text report from the violation database (*.avdb* file). The report contains the header, optional summary, and violation sections.

- The new `-relevance` argument is available for the following commands: `avdbreport`, `avdb2txt`, `avdb2html`, `avdb2csv`, `avdbclear`, and `avdbcopy`. This argument enables control over the violations while exporting a violation database to other formats. Depending on the setting, relevant, excluded, or all violations are considered.

  Note, that the default value for the reporting commands is `relevant`, which narrows down the number of reported violations and prints only the relevant ones. The `avdbclear` and `avdbcopy` commands process all violations by default.
- The new `avdb.exclusions.export` command enables exporting the exclusions from a violation database (*.avdb*) to *.txt* or *.csv* format. (SPT61850)
- The `-global_scope` argument of the `avdbcopy`, `avdbclear`, `avdbcompare`, `avdb2txt`, `avdb2csv`, and `avdb2html` macro commands no longer requires a file name specification. If present, this argument defines whether global scope violations should or should not be considered.
- The new `alint.policy.print` command enables exporting the information about rules used in a policy to *.txt* or *.csv* format. (SPT61234)
- The new console commands that enable setting global Preferences and design Properties are implemented. They work with options from the **Linting | General**, **Linting | Advanced**, and **Linting | Entries** pages. For more information, refer to the *Framework Basics* | *Design and Workspaces* | *Set Preferences in Command-line* section of the ALINT *User's Guide*. (SPT60841)
- The synthesis log (`-alint_synthesislog` option) is extended with the information about inferred flip-flops count. (SPT60567)
- The new `-alint_synthesispragmas` argument controls which rule checks to perform for code marked with synthesis constraints (pragmas). This argument is compilation-time only and available for `alog`, `acom`, and `alint` commands. It takes the following values: `none`, `syn` (by default), `all`.

  For example, the `-alint_synthesispragmas none` can be used to ignore the synthesis pragmas in files automatically generated by the Xilinx Core Generator. With pragmas ignored, the linting engine analyzes the source code enclosed by the pragmas. (SPT60662, SPT60815)
- The functionality of the `-relax` argument for `acom` has been extended to allow declaring entity attributes within an architecture of that entity. This functionality is required for compiling some models from selected vendor libraries. (SPT49762)
- The `acom` command now allows specifying the `-93`, `-2002`, or `-2008` arguments multiple times. Additionally, each argument can precede one source file or be followed by a number of sources. (SPT50344)
- The `-f` argument for the `acom` command can now be specified in the command line multiple times. (SPT50349)
- The error handling was improved for the `asim` command for cases where the object specified with the `-g` or `-G` argument cannot be found. In such cases, a warning message is printed to the console. (SPT51014)
- Now it is possible to control if additional directory is created for a new workspace or a design. The `-additional_dir` argument can be specified for the `workspace.create` or `workspace.design.create` console commands to enable or disable additional directory creation. (SPT60597)
- The rule level, severity, and origin name are now printed for the detail messages in the `avdb2csv` report. (SPT61085)

## Command-line Mode

- A new `alint.elabtimelinting` command starts elaboration-time linting of the design top-level unit in command-line mode. This command utilizes design and global preferences unlike `asim` and `alint` commands. The `alint.elabtimelinting` command is intended to be used with the `alint.compiletimelinting` command.

- It is now possible to specify a text file with arguments for the `vlint` stand-alone executable command (the `-f` option). This provides an ability to substitute several arguments specified directly on the command line.

### Documentation

- VHDL, Verilog, C++, and Do/Tcl code listings in the User Guide are now syntax-highlighted.

### Licensing

- If there are two or more license servers running, ALINT will check the licensing features from the first server. If all of the licenses are in use, the application will try to check the license from another server and so forth. Previously, ALINT always tried to check the license from the first server and only if it was turned off, it would try to fetch the license from another server. (SPT61266)
- Now the license queuing can be applied to the ALINT environment. This means that ALINT can be configured to repeat checking for a license with a specified time interval during the initialization. To enable license queuing at the application's initialization, the LICENSE_QUEUE environment variable must be set in the operating system. For details, refer to the *Installation and Licensing* | *Licensing* section of the ALINT User Guide.

### Miscellaneous

- The *system_info* script used on Linux for collecting information about the system where ALINT is running has been enhanced. You can run this script prior to contacting the support team. The file generated by the script will contain information about the CPU, memory, disk usage, etc. When launching the script, specify the ALINT installation directory as the first argument.

## Problems corrected in ALINT 2012.01

### Chip Level Engine

- Previously, a 2DFF synchronizer cell was recognized if it fed another flip-flop in the same clock domain. Now a line is considered synchronized even if there is no logic is after the 2DFF synchronizer cell. (SPT22055, SPT60303, SPT60779)
- Previously, the messages about the auto-detected global clock and reset signals were reported only if the related rules were a part of the active policy. The notification message is now printed regardless of the rules used. (SPT60750)
- Now global reset signals that are manually specified with the `-alint_grst` argument are displayed in the "Detected reset signals" section in the synthesis log. (SPT61104)
- Previously, a false violation was issued by the STARC_VHDL.2.5.1.5 rule checker, when a design unit in-out port was driven by a non-tri-state driver. (SPT60855)
- Performance was improved for the STARC_VHDL.1.3.1.6 and STARC_VHDL.3.3.6.2 rule checkers that check the mixture of clock and reset signal lines. (SPT61424)

- Previously, the global clock signal was not properly detected from the output ports of vendor library components with simplified handling.(SPT61984)
- Previously, the clock crossing was not detected when the data between two clock domains were transported through the `record` type. (SPT62316)

## Synthesis Emulation Engine

- Previously, false violations were issued by STARC_VHDL.3.3.3.2 due to incorrect synthesis of an enable chain if a default value assignment was present on the clock edge before an `if` statement. (SPT60856)
- Previously, unknown errors occurred when nested `if` statements were used within a process and a slice assignment was present. (SPT61360)
- Previously, a runtime exception occurred when a *synthesis.log* file was locked for writing and elaboration-time linting was invoked. (SPT61539)
- Previously, unknown error occurred when reordering was enabled and concurrent signal assignment statement that contained vector concatenation was used in the design. (SPT61486)

## VHDL Linting

- Previously, the false violations messages (ALDEC_VHDL.2051, STARC_VHDL.2.1.2.2, RMM.VHDL.5.2.1.4, DO254_VHDL.1314) were reported on built-in attributes. (SPT60628)
- Previously, a crash occurred when a flip-flop with an asynchronous reset was inferred for a two-dimensional array nested within a `for loop` construct. In the current version, the crash no longer occurs. (SPT60731)
- Now the output of the missing design unit can be specified as a global clock using the `-alint_gclk` argument. (SPT60306)
- Previously, unexpected compilation errors occasionally occurred on source files that contained more than 30000 lines. (SPT61221)
- An unhandled exception was issued after the data was restored for elaboration-time linting. (SPT61210)
- Previously, when a function call was used in a constant initial value assignment and this constant was used in a slice select, an unknown error occurred. (SPT61287)
- Previously, an error occurred when a multidimensional unconstrained array was assigned within a `loop` construct. (SPT61648)
- An issue was fixed where warnings about the incorrect order of files were printed although the compilation was launched in the automatic files ordering mode. (SPT51004)

## Verilog Linting

- Previously, the `translate_off` and `translate_on` synthesis constraints specified within the multi-line comments were not recognized by the ALINT engine. (SPT60627)
- The list of Verilog libraries specified in the preferences is correctly passed both to the compilation and elaboration stages. (SPT48759)

## Exclusions Files

- Previously, when the location of a file specified in the exclusions file was determined by a relative path, and was more than two directories up, the path information was processed incorrectly and so the file was

not found. (SPT61201)
- Previously, there was incorrect handling of the exclusions containing the basic-block related rules. (SPT61324)

## Configuration Viewer

- Previously, each time a policy or a ruleset file was saved, an order of internal elements in this file was randomly changed. Now all internal elements of a policy and a ruleset are sorted by name. (SPT60767)

## Flow Manager

- Now the folder name for the flow files consists of the flow name and *_flow_files* suffix (underlines instead of spaces). (SPT60768)
- Flow files structure was changed. Now the directories containing the phases data are stored directly in the *<flow_name>_flow_files* directory.
- Previously, elaboration-time linting was not invoked if the flow was used in console mode. (SPT60911)

## HDL Editor

- The issue with highlighting the matching `begin` - `end` keywords that enclose a nested `repeat` block was corrected. (SPT21708)
- The issue with collapsed sections of code being displayed incorrectly was resolved. The issue might occur when a new line was inserted immediately before a collapsed section. (SPT51251)

## Library Manager

- Previously, an empty warning was shown if a global library had been attached and the user did not have sufficient permissions to modify the global *library.cfg*. (SPT50855)

## AVDB Compare Dialog

- Previously, an incorrect comparison data base was generated if the comparison was invoked from the **AVDB Compare** dialog - regression and new violations were swapped. (SPT61479)

## Rule Plug-ins

### ALDEC

- The REGEXP_CONSTANT configuration parameter was added to the ALDEC_VHDL.2051 checker. It specifies the regular expression that constant names must comply with. (SPT61272)

- The false violations (ALDEC_VHDL.2201, ALDEC_VHDL.2208) were reported on the predefined attributes. (SPT60629)
- Previously, a false ALDEC_VHDL.2201 rule violation was issued on the object that was not considered as referenced when it had been used in a slice select. (SPT60709)
- Previously, a false ALDEC_VHDL.2201 rule violation was issued if an interface constant had been used in a bit-select of an array that contained objects of a record type. (SPT60903)
- Previously, a function was not considered as referenced when a function call had been present in a slice select, and so a false ALDEC_VHDL.2201 rule violation was issued. (SPT60780)
- The false violation was reported by the ALDEC_VHDL.2311 rule checker if a commented `process` statement was preceded with a statement that had a comment after it. (SPT60999)
- The ALDEC_VHDL.2351 rule checker was removed from the ALDEC_VHDL_DEFAULT policy because of the incorrect handling of the default values and multibit signals by this checker (see the Known Problems section of the Release Notes).
- Previously, the the ALDEC_VHDL.2051 rule checker reported false violations on the `generate` and `loop` variables while checking the constant names. (SPT61489)
- Previously, a false violation was reported by the ALDEC_VHDL.2305 rule checker on a `generate` block statement for which the generate condition was false. (SPT61209, SPT61482)
- Previously, a false violation was issued by ALDEC_VLOG.2202 rule checker on a continuous assignment. Now the violation is reported only on the variable initialization. (SPT61614)
- The LETTER_CASE configuration parameter has been added to ALDEC_VLOG.2052 checker. It provides a possibility to verify that port and module names contain lower case characters only. (SPT61239)
- Previously, false violations were issued by ALDEC_VHDL.2151 when loop variable was used in the slice select. (SPT61214)
- Previously, no violations were issued by ALDEC_VHDL.2201 when a signal was partially assigned in the port map. (SPT61692)

**STARC**

- Previously, a random instance name was printed in the STARC_VHDL.4.1.4.1 rule violation if a signal was assigned in the `always` statement and in the `task` that was used in the same `always` statement. (SPT60258)
- The performance was improved for the STARC_VLOG.4.1.4.1 rule checker. (SPT60294)
- Previously, a false STARC_VLOG.2.4.1.3 rule violation was reported when an unsized constant was present in the ternary expression in the signal initial assignment. (SPT60919)
- Previously, the line length was calculated incorrectly if literals were present so the STARC_VHDL.3.1.4.5 violation was printed. (SPT60812)
- Previously, no violations were present from STARC_VHDL.1.1.4.2 if lowercase letters were used in the constant name. (SPT60972)
- Previously, false violations were issued by STARC_VHDL.2.10.3.1 due to incorrect bit width calculation of the type conversion function that had an arithmetic expression as one of its arguments. (SPT61000)
- Previously, false violations were issued by STARC_VHDL.2.10.3.1 when the loop variables with different ranges were used in a conditional expression. (SPT61341)
- Previously, false violations were issued by STARC_VHDL.1.3.1.3 and STARC_VHDL.1.3.1.6 due to incorrect reset synchronizer detection. Now the reset synchronizer is not detected when different clocks lines are used for the first and second flip-flops of the reset synchronizer. (SPT61691)
- Previously, no violations were issued by STARC_VHDL.1.1.1.3 on `if` statement labels. (SPT60971)

### Batch mode

- Previously, when the linting process was invoked using *vlint.exe*, two AVDB files were generated. The compilation-time linting results were reported to the AVDB file, whose name was enclosed in braces: *{file_name.avdb}*. The name of the AVDB file with the elaboration-time linting results did not contain braces: *file_name.avdb*. Now the linting results for both stages are reported to the same AVDB file: *file_name.avdb*. (SPT61440)

### Documentation

- The description of the ALDEC_VHDL.2204 rule was corrected. Now the examples for this rule correspond to the checker behavior. (SPT60945)

## Known Problems

### General

- ALINT GUI for Linux does not work reliably over remote connections from Windows machines running the Hummingbird Exceed X Server version earlier than 13.0.14.542. This issue is not ALINT specific, it affects other software, including KDE applications running with the RENDER extension enabled.

  To work-around the issue, upgrade to Hummingbird Exceed version 13.0.14.542 or newer. If you do not have access to the latest Hummingbird Exceed version or do not want to upgrade, change the widget look in ALINT. This can be done by editing the *rungui* script in the installation directory and specifying the `-style` argument for the *alint* executable. The `-style` argument must be followed by the style name: `windows`, motif, `cleanlooks`, or `cde`. Do not use the `plastique` style. This style causes frequent errors in the X Server.
- ALINT GUI for Linux may encounter errors over a remote connection from Windows running X Window server delivered with Cygwin version earlier than 1.5.3.0 (released 2009.01.21). These errors are most likely to happen when a new window or a new document (e.g. a new waveform) is created or closed. This issue is not ALINT specific; it may affect all programs running remotely and be displayed on the X Window server delivered with Cygwin. To resolve the issue, upgrade to Cygwin version 1.7.1 or newer. (JKL1880)

### Chip Level Engine

- The `wait`-based FSMs are not supported by the ALINT engine. (SPT62055)

### VHDL Engine

- The number of violations issued by ALINT during the compilation- and elaboration-time linting to the Console may differ from the actual number of detected violations. It can occur if reordering is used in the VHDL compiler. The reason is that the design units are analyzed several times during reordering and so additional violations are produced. However ALINT correctly resolves the conflicts between them and

then stores the results to the AVDB file. Thus a violation database contains the correct set of violations while the Console contains all violations issued during the linting process. (SPT61525)

## Rule Plug-ins

### ALDEC

- The false violations from ALDEC_VHDL.2351 are reported if the default assignment is present. (SPT49312)
- The false violations from ALDEC_VHDL.2351 are reported if the different bits of an array are assigned. (SPT49311, SPT61242, SPT61770)

## Framework

- When document tabs do not fit into the document area, clicking a tab somewhere in the middle of the tab row may shift all tabs so that the clicked tab appears either on the left or on the right edge of the document area. (ARG4000)

## HTML Viewer

- If the selection of the **Use system proxy settings** option in the **Tools | HTML Viewer | Connection** category of the **Preferences** dialog box (**Tools | Preferences**) is required for any reason, and you encounter issues with browsing the Internet by using the HTML Viewer, an automatic detection of Internet settings in your system should be disabled. (ARG3714)

## Library Manager

- All libraries are collapsed each time the **Refresh** option is used. (MRP1816)

## File Browser

- The File Browser is not refreshed automatically after a new file is created. You need to switch focus to the browser or click the **Reload** button. (KAM2252)

## Macro Commands

- The script engine may take a long time (a couple of seconds) to detect that a given command does not exist. (PSL724)

## Reaching Technical Support

To open a Technical Support Case, users with a valid Maintenance Agreement should visit http://support.aldec.com/ContactSupport/. Please note, you will be asked to register if you have not already done so.