PEAR User Manual

Note: in this manual the directory in which PEAR has been installed is referred to with the `$PEAR_HOME` notation, while `$HOME` is the user's home directory.

Starting PEAR

PEAR can be run through a couple of scripts that may be found in the `$PEAR_HOME/bin` directory: the *Windows* version is `run.bat`, while `run.sh` is the *UNIX/Linux* one. Both scripts load the modules found in `$PEAR_HOME/lib` and set the *Java Virtual Machine* classpath so that any plug-in will be looked for in the `$PEAR_HOME/plugins` directory. PEAR's standard ouput and error are redirected into two log file that can be found in the `$HOME/.pear` directory.

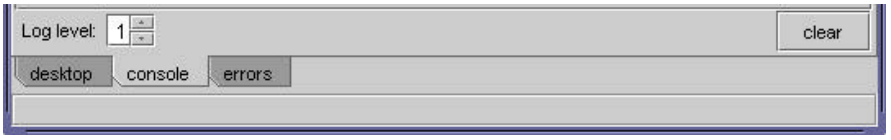
Graphic User Interface

PEAR provides a graphic interface whose windows are *dockable*, that is they can be attached to the four sides of the main desktop through the **Window** menu, which also provides the automatic window disposal by means of the **Dispose** command.

Through the **View** menu the hidden windows corresponding to the selected items can be shown or taken to the front; the **Validation Rule**s item is a submenu letting you choose whether show the actual window, or the inner elements: the source code of the validation rules, or the corresponding tree view.

The **Container** window shows the status of the containers and the substitution map during a validation session in step-by-step mode, **Console** shows the content of the log console, while **Protocol Source** holds a tab for each open protocol source.

Three tabs are placed below the desktop: the **desktop** tab is the one that has been described since now; **console** contains a text area showing the logger output and the controls you can use to clear the text area and set the logger actual verbosity; eventually the **errors** tab is displayed when an unhandled error is detected.

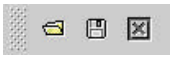


Below the tabs you can find the status bar, showing the last message produced by the tool, while above them the tool bars are placed.

The tool bars

Four tool bars are available:

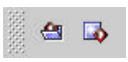
- the **file management** tool bar contains three buttons that let you open, save and close a file, respectively; typing the name of non-existing file in the "open file" dialog will cause a new one to be created, once saved;

- the **built-in resource browser** allows you to choose protocols or rule system included in the PEAR plug-in modules; by clicking the button on the left you can choose the type of resource to be opened, the other control lets you actually choose the resource; please note that you can not select any rule system if there are open protocol sources;

- the **validation rule** tool bar contains two buttons: the first one let you open a file containing a validation rule source; the second one sets as current the default validation rule system; the two buttons are active only when there are no open files, otherwise dangerous situations might happen;

- the **protocol management** tool bar provides four actions, respectively: show the warning window, switch between protocol displaying and editing mode, display the first error found if any, and open the tag inference dialog;

- the **validation** tool bar let you control the protocol validation process through the four buttons that provide the following actions respectively: go back to the previous validation step (if available), stop the protocol validation, run/complete the protocol validation, go to the next validation step; the following component is

used to set the validation step granularity, that is how often the process will stop to let the tool display its status: selecting the last level is the same as press the run button, the validation will complete; the log-level check-box allows you to link the logger verbosity to the validation step granularity: the tool will produce the appropriate amount of output in each moment, thus improving its performances.

The protocol source window

If you edit a protocol source and then you try to close it, an inner panel will be shown up asking if you wish to save the modified source.

An analogous element is shown up when the tag inference process is able to find one or more tagging suiting to the given protocol: a component appears in the inner panel allowing you to choose whether display the original source or the tagged ones.

The tag inference window

The tag inference window provides controls above and below the current source displaying area: the first control above allows you to choose whether display the original source or the taggings found; the **patience** slider allows you the select the number of equal taggings to be found before acting on the stack of backtrack nodes; the two following check-boxes enable or disable (if possible) the commit hints and the protocol reordering algorithm (it improves performances).

The panel placed below the source displaying area contains four buttons which provide the following actions respectively: run the tag inference process, stop it, save the current tagging as a separate file, close the tag inference window.

Configuration

Operation and container plug-ins don't require configuration: if you wish to add a jar module, just put it in the `$PEAR_HOME/plugins` directory; if you wish to add only a single class or resource, just place it in `$PEAR_HOME/plugins/it/unive/dsi/pear/plugins`. If you are running PEAR through WebStart, you can put your plug-in jar modules in `$HOME/.pear/plugins`. Please note that in this case you may provide only *resource* plugins.

Built-in resources may be specified trough two related list files named `builtin-vrules.ini` and `builtin-protocols.ini`, respectively. You can find the former in the `builtin.jar` module, while the latter is stored in the `protocols.jar` module. Once properly written the two lists, you'll have to make listed resources available by treating them as standard plugins. Alternatively you may provide no list, this way all the resources stored in any plugin module will be displayed by the resource browser.

^top