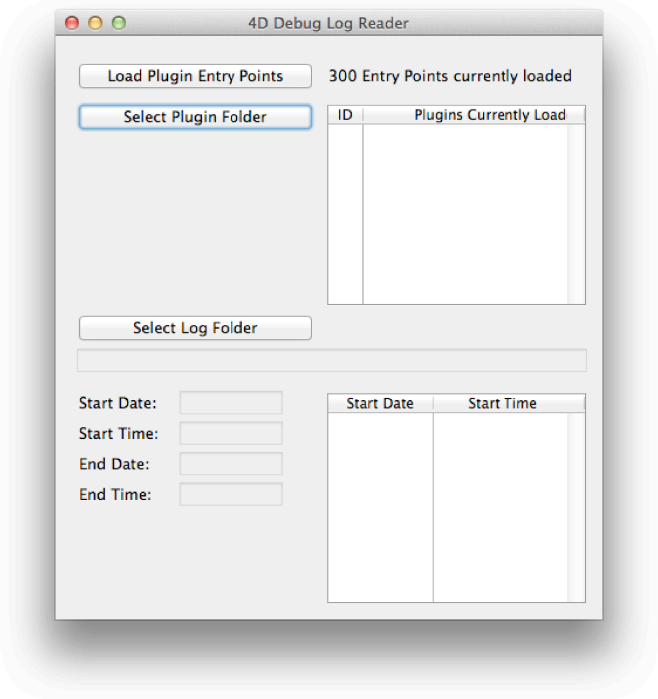
4D v14 Debug Log Reader

This application processes 4D debug log files created using the compact format introduced in 4D v14, i.e. with SET DATABASE PARAMETER(Debug log recording;2+4)

The application was written using 4Dv15R5, however the features it employs are available with any version of 4D from v15 onwards, on both Mac and Windows. It can also be compiled to improve performance and runs well using the new 64-bit Mono versions of 4D.

When first opening the application you will be prompted to create a new data file. This can be located anywhere.

The application opens to the Main window. This is done using the MAIN project method which is called from the On Startup database method.



In order to fully display the log entries in a human-readable format, two steps must first be performed:

- The 4D Plugin API entry points mapping between ID and name needs to be imported into the data file. This happens automatically when the application is first started.

  There is a file called EntryPoints.txt in the Resources folder which contains this name-to-ID mapping. It was generated from the EntryPoints.h header file in the 4D Plugin API.

  If the data file doesn't already contain the mappings, and the EntryPoints.txt file is missing from the Resources folder, the application will prompt for the EntryPoints.h header file. In this way the mappings can be updated when new versions of the 4D Plugin API become available.

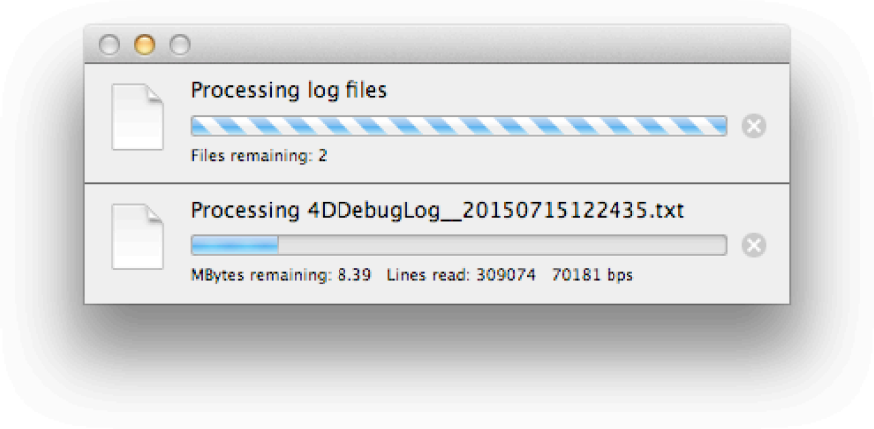
  The number of plugin entry points currently loaded is displayed on the main window.

- The plugins used by the application being debugged need to be parsed to extract the plugin command name-to-ID mapping. Only the plugin command IDs are recorded in the debug log so these mappings are used to convert those IDs to names for display.

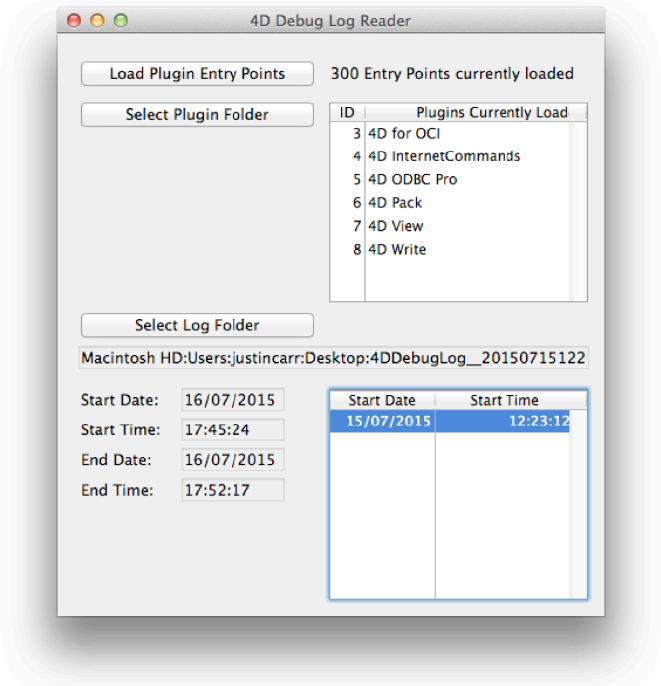
  This step is achieved by clicking the **Select Plugin Folder** button on the Main window and navigating to the Plugins folder used by the application being debugged.

  It is not a 100% reliable number-to-name mapping as it is not totally clear how it is performed by 4D. The mapping assumes that the first 2 IDs are used by 4D internally and this appears to be the case most of the time but not always. If the converted logs don't seem to be mapping the plugins correctly, the IDs can be changed by right-clicking on them in the list. The import will need to be performed again after changing any of these IDs.

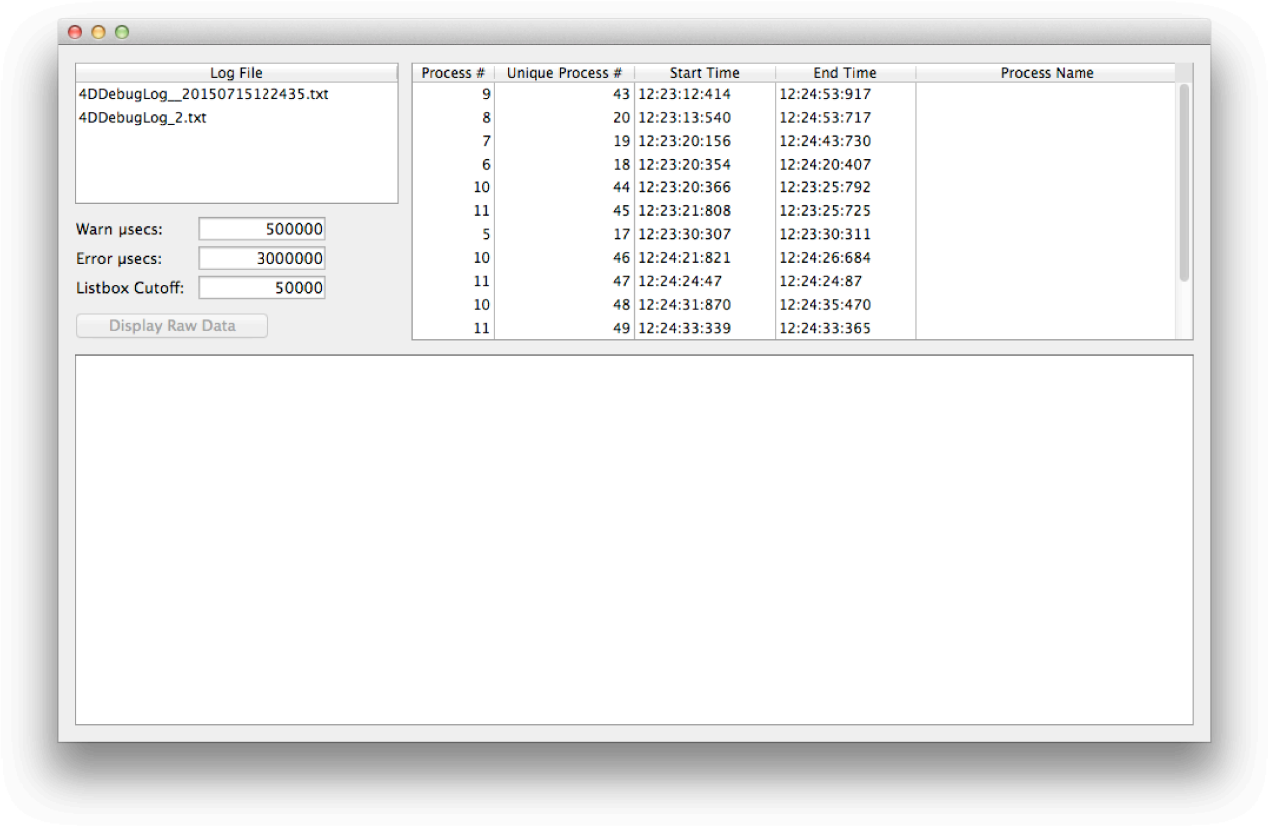
To parse the logs, click the **Select Log Folder** button and navigate to the folder containing the 4D debug log files. The import will begin automatically and progress is displayed.



Once the import has completed, the list of debugging sessions found within all of the files will be displayed. Each session is started and stopped with a call to SET DATABASE PARAMETER. A single debug log may contain multiple sessions, and a session may stretch across multiple files.



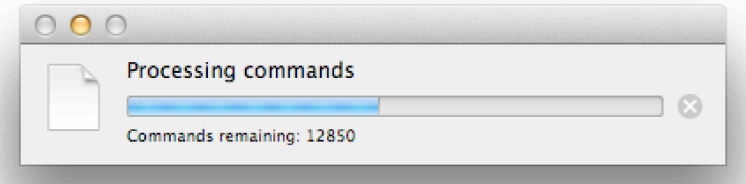
To view the contents of a session, double-click it in the list.



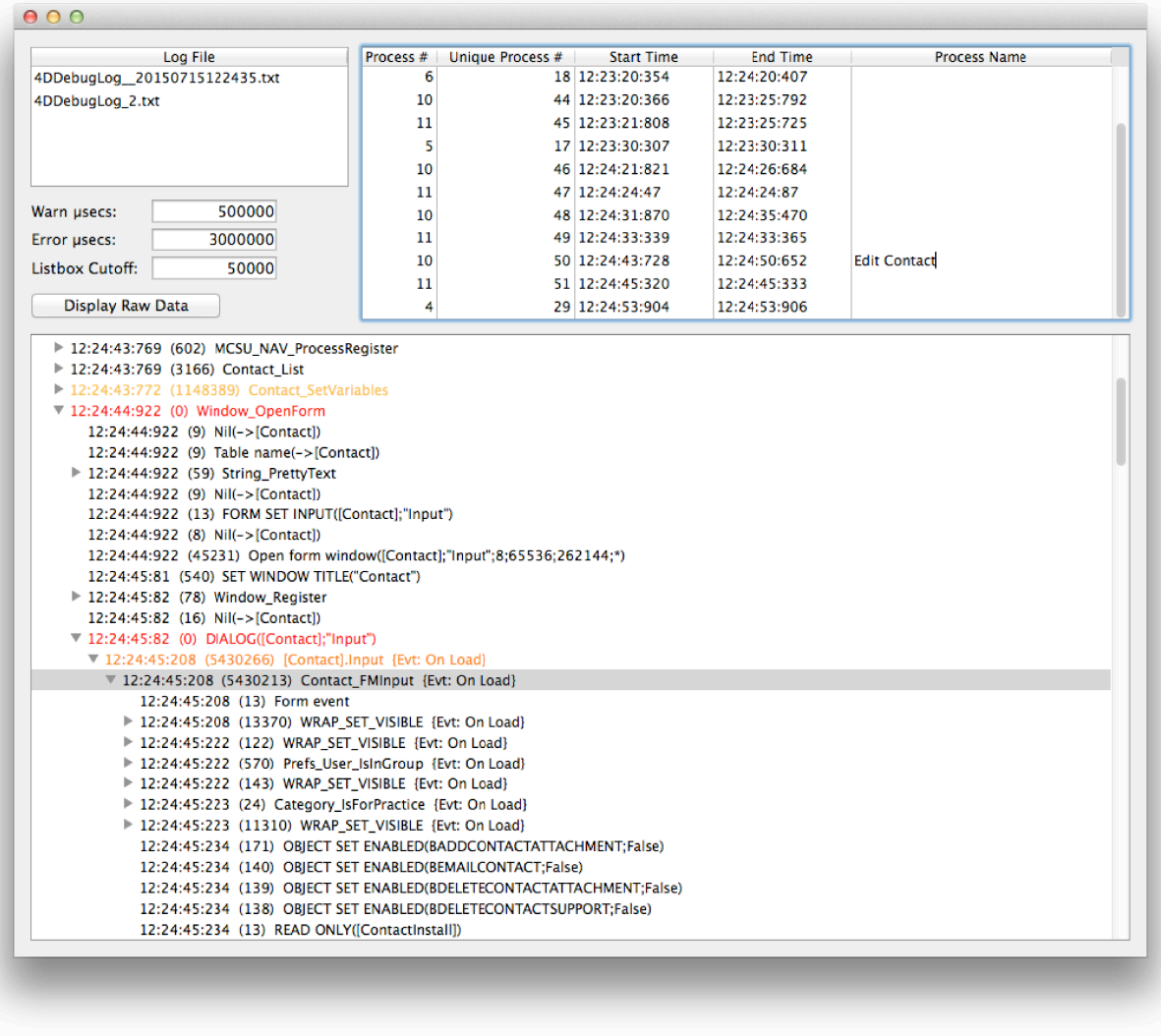
This window shows all of the processes contained within the displayed log files. Select one or more log files to narrow down to only the processes contained within those files.

The **Process Name** column is blank as the name can't be determined from the logs but it is an enterable listbox column so you can enter the process name yourself if you know what it is and wish it to be remembered.

Clicking on a process in the list will display the log entries for that process. If there are less than **Listbox Cutoff** entries for the process, they will be displayed in a hierarchical list, otherwise they will be displayed in a listbox. If the entries are going to be displayed in a hierarchical list, the first time a process is clicked a progress window will be displayed while the hierarchical list is built. Thereafter it is saved to the database so will display immediately.



Entries which take more than **Warn µsecs** microseconds to complete will be highlighted in orange. Those which take more than **Error µsecs** microseconds to complete will be highlighted in red.



Clicking the **Display Raw Data** button will display the records as they are saved in the database, without trying to render them to show the call stack.