Jekejeke Develop Installation

Version 1.0.9, September 9th, 2015

XLOG Technologies GmbH

Jekejeke Prolog

Development Environment 1.0.9

Installation Guide

Author: XLOG Technologies GmbH
Jan Burse
Freischützgasse 14
8004 Zürich
Switzerland

Date: September 9th, 2015
Version: 0.29

Participants: None

Warranty & Liability

To the extent permitted by applicable law and unless explicitly otherwise agreed upon, XLOG Technologies GmbH makes no warranties regarding the provided information. XLOG Technologies GmbH assumes no liability that any problems might be solved with the information provided by XLOG Technologies GmbH.

Rights & License

Trademarks

Table of Contents

Change History

Jan Burse, June 9th, 2010, 0.1:
- Initial version.

Jan Burse, July 2nd, 2010, 0.2:
- Code styling introduced.

Jan Burse, July 25th, 2010, 0.3:
- Release notes for 0.8.4 added.

Jan Burse, October 2nd, 2010, 0.4:
- Release notes for 0.8.5 added.

Jan Burse, November 13th, 2010, 0.5:
- Release notes 0.8.6 new decimals, references, number syntax and string syntax.

Jan Burse, December 2nd, 2010, 0.6:
- Release notes 0.8.6 new character, string and stream control predicates.

Jan Burse, January 2nd, 2011, 0.7:
- Release notes for 0.8.7 added.

Jan Burse, April 15th, 2011, 0.8:
- Release notes for 0.8.8 added and separation application and documentation.

Jan Burse, April 17th, 2011, 0.9:
- Runtime library installation guide moved into separate document.

Jan Burse, Mai 6th, 2011, 0.10:
- Release notes for 0.8.9 added and documentation platform independent.

Jan Burse, June 6th, 2011, 0.11:
- Release notes for 0.9.0 added and samples section included.

Jan Burse, September 27th, 2011, 0.12:
- Release notes for 0.9.1 added.

Jan Burse, November 15th, 2011, 0.13:
- Release notes for 0.9.2 added.

Jan Burse, February 23th, 2012, 0.14:
- Release notes for 0.9.3 added.

Jan Burse, June 4th, 2012, 0.15:
- Release notes for 0.9.4 added.

Jan Burse, August 30th, 2012, 0.16:
- Release notes for 0.9.5 added.

Jan Burse, December 10th, 2012, 0.17:
- Release notes for 0.9.7 added.

Jan Burse, February 20th, 2013, 0.18:
- Release notes for 0.9.8 added and system requirements sections move.

Jan Burse, April 7th, 2013, 0.19:
- Release notes for 0.9.9 added.

Jan Burse, July 27th, 2013, 0.20:
- Release notes for 0.9.10 added.

Jan Burse, August 14th, 2013, 0.21:
- Known issues sections moved.

Jan Burse, December 6th, 2013, 0.22:
- Release notes for 0.9.12 added and automatic discovery introduced.

Jan Burse, April 1st, 2014, 0.23:
- Release notes for 1.0.1 added.

Jan Burse, July 23rd, 2014, 0.24:
- Release notes for 1.0.3 added and known issues section expanded.

Jan Burse, August 17th, 2014, 0.25:
- Release notes for 1.0.4 added and known issues section expanded.

Jan Burse, February 27th, 2015, 0.26:
- Release notes for 1.0.5 added.

Jan Burse, May 28th, 2015, 0.27:
- Release notes for 1.0.6 added.

Jan Burse, July 3rd, 2015, 0.28:
- Release notes for 1.0.7 added.

Jan Burse, September 9th, 2015, 0.29:
- Release notes for 1.0.9 added.

1 Introduction

The Jekejeke Prolog development environment is available for Windows, Linux and Macintosh. Customers can also download documentation and samples. In the following we describe the download contents and its most basic use.

- **Release Notes:** This section lists the changes concerning the Jekejeke Prolog development environment, documentation and samples.

- **Environment Installation:** Here we describe how the Jekejeke Prolog development environment can be installed and executed.

- **Support Files:** Here we describe how the Jekejeke Prolog development environment support files can be installed.

2 Release 1.0

This section lists the changes concerning the console manual, the language reference and the programming interface between the different releases. The following releases have been made available so far:

2.1 Release 1.0.9

The following features have been provided for the Jekejeke Prolog development environment of version 1.0.9:

Language

- The pseudo module inspection/direct removed.
- New module inspection/syntax introduced.
- New module inspection/provable introduced.
- New predicate property sys_noinstrument introduced.
- .

2.2 Release 1.0.7

The following features have been provided for the Jekejeke Prolog development environment of version 1.0.7:

Language

- Predicate property sys_spy_point/0 removed.
- Clause property sys_context/1 removed.
- Clause property source_file/1 removed.
- Clause property line_no/1 removed.
- Prolog flag sys_trace removed.
- Prolog flag source_file removed.
- Prolog flag line_no removed.
- Predicate notrace/0 removed.
- Predicate debugging/1 removed.
- New values for Prolog flag debug introduced.
- New predicate skip/0 introduced.
- New predicate leash/1 introduced.
- New predicates pin/1 and nopin/1 introduced.
- New module system/protocol introduced.
- The debugger prompt is now a little REPL.
- New predicate out/0 introduced.

Swing Interface

- New debug toolbar introduced.
- New debug menu introduced.

Native Console

- Ctrl-C does not provide a character menu anymore.
- Ctrl-C directly leads to a pause and hence the debugger REPL.

2.3 Release 1.0.6

The following features have been provided for the Jekejeke Prolog development environment of version 1.0.6:

Language

- Error messages removed from documentation.
- Error messages are now multi-lingual, English and German.
- Error messages are now found on open source web site.
- The documentation has now a title page.

2.4 Release 1.0.5

The following features have been provided for the Jekejeke Prolog development environment of version 1.0.5:

Language

- The clause property sys_notrace removed.
- The frame property sys_witnesses removed.
- The stack property sys_stack_frame removed.
- The stack property sys_call_indicator removed.
- The stack property sys_text_frame removed.
- The frame property sys_bindings removed.
- New frame property variables introduced.

Swing Interface

- Icon scaling on high-res displays.
- Table scaling on high-res displays.

2.5 Release 1.0.4

The following features have been provided for the Jekejeke Prolog development environment of version 1.0.4:

Language

- Filler annotation option causes Prolog reader to capture fillers.
- Navigation format option causes Prolog writer to emit navigation comments.
- Filler annotation option causes Prolog write to emit fillers.
- .

2.6 Release 1.0.3

The following features have been provided for the Jekejeke Prolog development environment of version 1.0.3:

Language

- New predicate listing/2 introduced.
- New predicate syntax_property/2 introduced.
- New predicate set_syntax_property/2 introduced.
- New predicate reset_syntax_property/2 introduced.

2.7 Release 1.0.1

The following features have been provided for the Jekejeke Prolog development environment of version 1.0.1:

Language

- New predicate direct_property/2 introduced.
- New predicates set_direct_property/2 and reset_direct_property/2 introduced.
- New predicate debugging/1 introduced.
- The predicate debugging/1 now understands the colon (:)/2.
- The predicate friendly/1 now understands the colon (:)/2.
- The predicate dump/1 now understands the colon (:)/2.

3 Release 0.9

This section lists the changes concerning the console manual, the language reference and the programming interface between the different releases. The following releases have been made available so far:

3.1 Release 0.9.12

The following features have been provided for the Jekejeke Prolog development environment of version 0.9.12:

Language

- New command line option -a introduced.

3.2 Release 0.9.10

The following features have been provided for the Jekejeke Prolog development environment of version 0.9.10:

Language

- The predicates friendly/0 and friendly/1 now respect public/private.
- The predicates dump/0 and dump/1 now respect public/private.

3.3 Release 0.9.9

The following features have been provided for the Jekejeke Prolog development environment of version 0.9.9:

Language

- The predicate expand_goal/2 is not anymore traced for top-level queries.
- The predicate expand_term/2 is not anymore traced for consulting.
- The predicate sys_rebuild_term/2 is not anymore traced for listing.
- New clause property sys_notrace/0 introduced.

3.4 Release 0.9.8

The following features have been provided for the Jekejeke Prolog development environment of version 0.9.8:

Language

- The predicate sys_ignore/1 now uses Java body conversion.

3.5 Release 0.9.7

The following features have been provided for the Jekejeke Prolog development environment of version 0.9.7:

Language

- New system predicates expose_goal/2 and goal_exposing/2 introduced.

3.6 Release 0.9.5

The following features have been provided for the Jekejeke Prolog development environment of version 0.9.5:

Android Interface

- New menu item trace introduced.

3.7 Release 0.9.4

The following features have been provided for the Jekejeke Prolog development environment of version 0.9.4:

Swing Interface

- Menu item abort has now period accelerator.
- The comma accelerator now circles between debug, trace and no-debug.
- Find command now defaults to the selected text.

Language

- New system predicate asserta/2 and assertz/2 introduced.
- New system predicate clause/3 and retract/2 introduced.
- New break debugger command to enter query answer loop introduced.
- New intermediate form instruction last_display introduced.

Programming Interface

- CapabilityEnvironment moved to package jekdev.platform.headless.
- ToolkitEnvironment moved to package jekdev.platform.headless.

3.8 Release 0.9.3

The following features have been provided for the Jekejeke Prolog development environment of version 0.9.3:

Swing Interface

- Now possible to start a thread in a new tab.

Language

- Now possible to break into the debugger at any time. (Finding U. Neumerkel)

Programming Interface

- New class ToolkitEnvironment introduced.
- New class CapabilityDevelopment introduced.

3.9 Release 0.9.2

The following features have been provided for the Jekejeke Prolog development environment of version 0.9.2:

User

- New debugger command “b” for bindings introduced.
- New text panel in settings dialog introduced.

Language

- New system predicates dump/0 and dump/1 introduced.
- New system predicate compile_term/3 introduced.
- New predicates sys_recordz_clause/1 and sys_recorda_clause/1 introduced.
- New system predicate sys_erase_clause/1 introduced.
- New system predicate clause_property/2 introduced.
- New predicates set_clause_property/2 and reset_clause_property/2 introduced.
- New system predicate clause_term/3 introduced.
- New system predicate sys_instance_clause/2 introduced.
- New system predicate frame_property/2 introduced.
- New predicates set_frame_property/2 and reset_frame_property/2 introduced.
- New call_meta and last_meta intermediate codes introduced.
- New system predicates trace_goal/2 and goal_tracing/2 introduced.
- New system predicates sys_prepare_body/2 and sys_unfold_body/1 introduced.
- New Prolog flag sys_query_frame introduced.
- New Prolog flag sys_skip_frame introduced.
- New Prolog flag sys_cloak introduced.
- New system predicate sys_ignore/1 introduced.

3.10 Release 0.9.1

The following features have been provided for the Jekejeke Prolog development environment of version 0.9.1:

Console

- Colour map with lighter and darker colours now used.
- Selection colours can now be configured as well.
- New menu item “Kill” to forcefully stop a Prolog thread.
- New menu item “Debug” to toggle the debug mode.
- New edit panel item “Debug” to toggle the debug mode.

Language

- New Prolog flag sys_clause_instrument introduced.
- New instructions “unify_atomic” and “unify_compound” introduced.

3.11 Release 0.9

The following features have been provided for the Jekejeke Prolog development environment of version 0.9:

Console

- Text pane does not loose style anymore when selecting input start.
- Text pane does not allow edit before input start.
- Text pane positioning problem during search solved.
- Can now change background color in terminal settings.
- Can now change input and output color in terminal settings.
- Ctrl-H key blocked since it would delete character to the left before input start.
- Ctrl-Backward key blocked since it would delete word to the left before input start.
- Can now toggle trace from tool bar.
- Toggle trace has now accelerator key Ctrl-Period.

Language

- New system predicate break/0 introduced.
- New instructions “last_goal” and “last_cont” now shown in intermediate form.
- Instruction “unify_term” inside body now shown in intermediate form.
- New Prolog flag disp_input introduced.
- New Prolog flag disp_output introduced.

4 Release 0.8

This section lists the changes concerning the console manual, the language reference and the programming interface between the different releases. The following releases have been made available so far:

4.1 Release 0.8.9

The following features have been provided for the Jekejeke Prolog development environment of version 0.8.9:

Console

- Look and feel change also updates text pop up menu now.
- New dialog for activation over service.
- New dialog for activation over email.
- New dialog for enlisting start-up capabilities.

Language

- Prolog flag source_file moved from runtime library to here.
- Prolog flag line_no moved from runtime library to here.
- System predicate sys_clause/3 moved from runtime library to here.
- System predicate sys_retract/2 moved from runtime library to here.
- System predicate sys_assertz/2 moved from runtime library to here.
- System predicate sys_asserta/2 moved from runtime library to here.
- New system predicate sys_compile/3.
- New system predicate sys_instance/3.
- Better call site provisioning in query answer loop.
- Better call site detection in debugger.
- Better sys_notrace handling in debugger.
- New t (Text) debugger command introduced.
- New g (Ancestors) debugger command introduced.

4.2 Release 0.8.8

The following features have been provided for the Jekejeke Prolog development environment of version 0.8.8:

Console

- Query answer loop now suppresses display of system exceptions.
- Query answer loop now reacts on user exit by leaving the answer loop.
- Query answer loop now reacts on all other system exceptions by leaving all loops.
- End of file (^D) during query answer now acts as a user exit.
- The “w” during query answer now acts as a user close.
- End of file (^D) during trace message now acts a user exit.
- The “w” during trace message now acts as a user close.
- New predicates abort/0, exit/0 and close/0 introduced.

4.3 Release 0.8.7

The following features have been provided for the Jekejeke Prolog development environment of version 0.8.7:

Console

- Thread tour completed.
- Trace menu item can now toggle the trace mode.
- Abort menu item does now also interrupt threads.
- Hold menu item is available independent of thread state.
- Signals and interrupts are now cleared in the top level.
- Memory low tour completed.

4.4 Release 0.8.6

The following features have been provided for the Jekejeke Prolog development environment of version 0.8.6:

Console

- Fixed problem dispose of running window when user cancelled operation.
- Fixed locking problem during key repeat.
- Fixed end of line detection problem.
- Console is now styled with different styles for input and output.
- Improved menu handling on the Macintosh platform.

4.5 Release 0.8.5

The following features have been provided for the Jekejeke Prolog development environment of version 0.8.5:

Console

- New view menu items “Hold Screen” and “Clear Screen”.
- New view menu items “Submit Input” and “Submit Eof”.
- New search menu items “History up” and “History down”.
- New terminal panel in settings dialog.
- Help button in register and settings dialog.
- Query variable names in answer and debug message.
- Clause variables names in listing and friendly.

4.6 Release 0.8.4

The following features have been provided for the Jekejeke Prolog development environment of version 0.8.4:

Console

- Trace and abort menu item disabled during start-up.
- Menu item for the installation guide.

4.7 Release 0.8.3

We started the beta-testing campaign with this release. This is our first public release.

5 Environment Installation

The environment comes in two flavours. There is a version for the Swing Java virtual machine and a version for the Android Java virtual machine.

The different flavours do not deliver exactly the same functionality. The environment inherits the differences already found in the library. Currently the environment does not add some major additional differences. For a summary of the differences found in the library see the library installation documentation.

The code of the libraries is also mostly identical except for some places where different system packages are used. Because of these differences the Android library will not run in a Swing virtual machine, and the Swing library will not run in an Android virtual machine.

In the following we will give more details on the installation of the different versions:

5.1 Swing Installation

A manual package is available for any Swing user interface toolkit implementation. The package includes the Jekejeke Prolog development environment. You will download the following archive file:

```
toplevel.jar # Top-Level and Embedding
```

You might copy the archive file to the destination directory `<dest>` of your choice. The archive file can be used for the following purposes:

- **Activation:** The archive file can be used to activate licenses.
- **Top-Level:** The archive file can be used to execute a Prolog query answer loop.
- **Embedding:** The archive file can be embedded into Swing applications.
- **Automatic Discovery:** The archive file discovers class paths and capabilities.
- **System Requirements:** The system requirements of the actual version.

Activation

The archive file can be used to activate licenses. The environment archive file itself does need a license and additional capabilities might need a license. The activation can be done either with the original archive file or when the archive file has been unpacked and included in a new archive file. The following methods are available for activation:

- **Non-Graphical License Manager:** The non-graphical top level automatically queries the end-user via prompt lines for the activation of licenses. The subsequent top-level section provides more information on how to start the non-graphical top-level.
- **Graphical License Manager:** The graphical top level automatically queries the end-user via dialog windows for the activation of licenses. The subsequent top-level section provides more information on how to start the graphical top-level.
- **Custom License Manager:** Applications and libraries that embed the archive file can code their own license management interactions. The subsequent embedding section provides more information on building applications and libraries.

Top-Level

The archive file can be used to execute a Prolog query answer loop. You will need a Java runtime environment so that you have a java command available. The following command will then execute the archive file and start the Prolog query answer loop without a graphical user interface.

```
java <options> -jar toplevel.jar -h <arguments>
```

To start the Prolog query answer loop with a graphical user interface the following command can be used. On Windows one might also use javaw instead of java:

```
java <options> -jar toplevel.jar <arguments>
```

The following options are recommended:

```
-Duser.language=<language_code>     # Locale language
-Duser.region=<country/area code>   # Locale country
-mx<size>                           # Available memory
-Dapple.laf.useScreenMenuBar=true   # On Mac OS only.
-Dapple.awt.brushMetalLook=true     # On Mac OS only.
-Xdock:name=Jekejeke                # On Mac OS only.
```

Alternatively one can also double click the `toplevel.jar` which executes the archive file with the current default Java runtime and without any options or arguments. This works mostly for Windows and Mac OS, but might fail on Linux.

The archive file accepts further arguments. A detailed documentation of the archive file arguments can be found in the programming interface document for the class ToolkitEnvironment.

There is no need to unpack the archive file. Avoid re-deploying once the archive file has been activated, since this might invalidate the activation.

Embedding

The archive file can be embedded into variety of Swing applications. Let's look at the case of an embedding inside a Java standalone application. Assume that your Java class `<main>` has a static method main() and that it resides inside the destination directory `<dest>`. Assume further that this class will use the Hotspot runtime library of Jekejeke Prolog.

You will first need a Java development kit so that you have a Java compiler available. Your Java class `<main>` can be compiled by the following command form the destination directory `<dest>`. Note the different path separators on the different platforms:

```
javac -cp toplevel.jar;. <main>.java    # on windows
javac -cp toplevel.jar:. <main>.java    # on linux and mac
```

You will then need a Java runtime environment so that you have a Java runtime available. Your Java class <main> can be executed by the following command from the destination directory <dest>. Note again the different path separators on the different platforms:

```
java -cp toplevel.jar;. <main>          # on windows
java -cp toplevel.jar:. <main>          # on linux and mac
```

Alternatively you can use an integrated development environment to compile and execute your Java class. All you probably have to do is create an appropriate project and then register the archive file of the Jekejeke Prolog runtime library in the class path of the project.

You might also unpack the Jekejeke Prolog development environment and include it in a .jar together with your compiled byte code and then execute this .jar.

Further you might want to deploy the Jekejeke Prolog runtime library together with your applets or servlets. In the case of applets all you need to do is mention the archive file in the applet tag and copy the archive file to the web server together with the applet. In case of servlets all you need to do is copy the archive file into the WEB-INF/lib directory. For more details see the deployment study document.

Avoid changing the destination directory once the new archive file has been activated, since this might invalidate the activation.

Automatic Discovery

Since release 0.9.12 of the Jekejeke Development Environment we have facilitated the selection of class paths. This feature is only available for the graphic invocation of the Jekejeke Development Environment. Upon start-up the interpreter will first check the following directory for additional class path elements:

```
    <working directory>/apk
```

If this directory contains class path elements which are not yet listed in the class path settings or which have not yet been added by the new -a command line option a graphical dialog will be shown to the end-user. The end-user can then decide which additional class path elements from the above directory should be included or excluded.

Since release 0.9.12 of the Jekejeke Development Environment we have also facilitated the selection of capabilities. This feature is also only available for the graphic invocation of the Jekejeke Development Environment. Upon start-up of the interpreter and when the class path elements have been registered, the class path elements are search for package slips.

If the package slips contain capabilities which are not yet listed in the capabilities settings or which have not yet been added by the -e command line option a graphical dialog will be shown to the end-user. The end-user can then decide which additional capabilities from the package slips should be included or excluded.

System Requirements

The Jekejeke Prolog development environment of version 0.9.8 requires at least:

Graphic interface

- Swing 1.6 [1]

Headless

- Hotspot 1.5 [4]

The Jekejeke Prolog development environment of version 0.9.8 is compatible with at least:

- Jekejeke Prolog Runtime Library 0.9.8, Swing Version [2]

5.2 Android Installation

Manual packages are available for any Android Java virtual machine. The packages only include the Jekejeke Prolog development environment. You will download the following archive files:

```
toplevel.apk # Top-Level
toplevel.zip # Embedding
```

You might copy the archive file to the destination directory `<dest>` of your choice. The archive file can be used for the following purposes:

- **Activation:** The archive file can be used to activate licenses.
- **Top-Level:** The archive file can be used to execute a Prolog query answer loop.
- **Embedding:** The archive file can be embedded into Android applications.
- **Automatic Discovery:** The archive file discovers class paths and capabilities.
- **System Requirements:** The system requirements of the actual version.

Activation

The archive file can be used to activate licenses. The environment archive file itself does need a license and additional capabilities might need a license. The activation can be done either with the original archive file or when the archive file has been unpacked and included in a new archive file. The following methods are available for activation:

- **Graphical License Manager:** The graphical top level automatically queries the end-user via dialog windows for the activation of licenses. The subsequent top-level section provides more information on how to start the graphical top-level.
- **Custom License Manager:** Applications and libraries that embed the archive file can code their other license management interactions. The subsequent embedding section provides more information on building applications and libraries.

Top-Level

The archive file can be used to execute a Prolog query answer loop. The archive file can either directly or indirectly be deployed on a device.

For direct deployment change the application preferences on your device to allow download from arbitrary locations. Open a browser on the device and then navigate to the download page of our sales system. Some devices might work better when the sales system is browsed without frames. Finally click on the corresponding download link. This will initiate first a local download and then a local deployment of the archive file on the device.

For indirect deployment you might copy the archive file to the destination directory `<dest>` of your choice and then remotely deploy it to a device. You will then need an Android development kit so that you have a deployment tool. The following step might then do the remote deployment:

- **adb:** Install your Android package on a device.

The above works for an Android device connected via USB or for an Android emulator present on the download platform.

There is no need to unpack the archive file. Avoid re-deploying once the archive file has been activated, since this might invalidate the activation.

Embedding

The archive file can be embedded into variety of Dalvik applications. Let's look at the case of an embedding inside an Android activity. Assume that your Java class `<activity>` derives from the class android.app.Activity and that it resides inside the destination directory `<dest>`. Assume further that this class will use the Dalvik runtime library of Jekejeke Prolog. Further assume that we do cross compilation on a traditional Java platform for an Android emulator or a remote Android device.

You will first need a Java development kit so that you have a Java compiler available. You will also need the Android development kit so that the Android libraries are available. Before you can start compiling your classes the following step might be necessary:

- **aapt:** Compile the manifest and your Android resources.
- **aidl:** Compile your Android interface definitions.

Your Java class `<activity>` can be compiled by the following command form the destination directory `<dest>`. Note the different path separators on the different platforms:

```
# on windows
javac -bootclasspath android.jar \
      -cp toplevel.zip;. \
      <activity>.java
# on linux and mac
javac -bootclasspath android.jar \
      -cp toplevel.zip:. \
      <activity>.java
```

Further steps that are necessary in the process of building an Android package are:

- **dex:** Convert the class files to Dalvik byte code.
- **apkbuilder:** Create an Android package.
- **Jarsigner:** Sign the Android package.
- **zipalign:** Align the Android package.
- **adb:** Install your Android package on a device.

Alternatively you can use an integrated development environment to compile and execute your Java class. All you probably have to do is create an appropriate project and then register the archive file of the Jekejeke Prolog runtime library in the class path of the project. The development environment might invoke the installation for you.

The above works for an Android device connected via USB or for an Android emulator started on the development platform. Alternatively you can upload your Android package to an internet store or to an internet site. Then point your device to the internet store or to the internet site to launch the package.

Automatic Discovery

Since release 0.9.12 of the Jekejeke Development Environment we have facilitated the selection of class paths. Upon start-up the interpreter will first check the Android package manager for additional class path elements:

```
    packages with the same user id
```

If this list contains class path elements which are not yet listed in the class path settings a graphical dialog will be shown to the end-user. The end-user can then decide which additional class path elements from the above directory should be included or excluded.

Since release 0.9.12 of the Jekejeke Development Environment we have also facilitated the selection of capabilities. Upon start-up of the interpreter and when the class path elements have been registered, the class path elements are search for package slips.

If the package slips contain capabilities which are not yet listed in the capabilities settings a graphical dialog will be shown to the end-user. The end-user can then decide which additional capabilities from the package slips should be included or excluded.

System Requirements

The Jekejeke Prolog development environment of version 0.9.8 requires at least:

Graphic interface

- Android 2.2 (API 8) [6]

Headless

- Dalvik 1.6 (API 4) [5]

The Jekejeke Prolog development environment of version 0.9.8 is compatible with at least:

- Jekejeke Prolog Runtime Library 0.9.8, Android Version [2]

6 Support Files

Download of the support files is available for all platforms that have a ZIP extractor. The download includes the support files for the Jekejeke Prolog runtime library and the Jekejeke Prolog development environment. You will download the following archive file:

```
supdev.zip        # The support files archive
```

You can use a GUI tool or a commend line tool of your choice that is able to deal with .zip files. If all else fails you can use the jar utility that comes with a Java development kit installation. The archive file can be extracted with the following jar utility command. Make sure that you are inside destination directory `<dest>`:

```
jar xf supdev.zip
```

After unpacking the archive one can easily explore its contents with a HTML browser. It is also possible to browse the documentation from within the development environment via the help menu item. Just see to it that the destination directory `<dest>` matches the base URL of the development environment.

The support files archive contains the following kind of support files:

- **Documentation:** The documentation for the development environment is provided as HTML split files or as full PDF documents.
- **Example Sources:** The source files for the development environment example programs are provided as ZIP archive files.
- **Interpreter Sources:** The partial source files for the development environment interpreter are provided as a ZIP archive files.

6.1 Documentation

The documentation for the runtime library is provided as HTML split files or as full PDF documents. The HTML split files can be view with a HTML browser. To view the PDF files a PDF reader needs to be available.

The support files archive contains the following documentation:

```
05_run            # Runtime Library
 +--- 10_docu            # Documentation
 |     +--- 00_android        # User Manual Android
 |     +--- 01_swing          # User Manual Swing
 |     +--- 02_reference      # Language Reference
 |     +--- 03_interface      # Programming Interface
 |     +--- 04_installation   # Installation Guide
 |     +--- 05_frequent       # Frequent Predicates
 +--- 15_stdy           # Studies
       +--- 06_bench          # Benchmark Results
       +--- 08_deploy         # Deployment Methods
       +--- 07_compliance     # Compliance Results
10_dev            # Development Environment
 +--- 10_docu           # Documentation
       +--- 00_android        # User Manual Android
       +--- 01_swing          # User Manual Swing
       +--- 02_reference      # Language Reference
       +--- 03_interface      # Programming Interface
       +--- 04_installation   # Installation Guide
```

The full PDF documents are located in the files called package.pdf in the above directories.

6.2 Example Sources

The source files for the development environment example programs are provided as source archive files. The source files mainly include Prolog texts and Java classes. But they might also include other types of artefacts.

The support files archive contains the following sources:

```
05_run            # Runtime Library
 +--- 10_docu           # Documentation
 |     +--- 01_swing          # User Manual Swing
 |     +--- 02_reference      # Language Reference
 |     +--- 03_interface      # Programming Interface
 |     +--- 05_frequent       # Frequent Predicates
 +--- 15_stdy           # Studies
       +--- 06_bench          # Benchmark Results
       +--- 08_deploy         # Deployment Methods
       +--- 07_compliance     # Compliance Results
10_dev            # Development Environment
 +--- 10_docu           # Documentation
       +--- 02_reference      # Language Reference
       +--- 03_interface      # Programming Interface
```

The source archive files are located in the files package.zip in the above directories.

You can easily run the programs by means of the Java command line or from within an integrated development environment. Some programs from the deployment methods document demand a web server, an SQL database, a HTML browser or an applet runner for execution. For more details see the corresponding documentation.

6.3 Interpreter Sources

The partial source files for the development environment interpreter programs are provided as source archive files. The source files mainly include Prolog texts and Java classes. But they might also include other types of artefacts.

The support files archive contains the following sources:

```
05_run                 # Runtime Library
 +--- 02_reference           # Language Reference
 +--- 05_frequent            # Frequent Predicates
```

The source archive files are located in the files package.zip in the above directories.

The sources are mainly there to give a more detailed documentation of the inner working of the interpreter. But the sources can also be used to create derivative work, except for special predicates, which currently come without source.

7 Known Issues

The following issues are known for the Jekejeke Prolog development environment of version 1.0.7:

Language

- Navigation for head predicates should be web site search.
- Should have a meta-interpreter flag for predicates.
- If the meta-interpreter flag is set the trace should check the first argument.
- There is not yet a read/clause option for sub goal positions.
- Should make use of determinism check to reduce shown debugger ports.
- Should make use of clean-up facility to provide a debugger exception port.
- During abort cutting the debug/trace mode should be temporarily switched off.
- The debugger prompt REPL should do goal expansion.
- Should have predicates to compensate for removed debugger commands.

Swing

- Some problems with the debugging accelerator keys in JDK 1.7 on Mac OS.

Pictures

Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.

Tables

Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.

References

[1] Java 2 Platform Standard Edition 6.0, Mustang, Sun Microsystems, 2006
http://www.oracle.com/technetwork/java/javase/overview/index-jsp-136246.html

[2] Prolog Runtime Library, XLOG Technologies GmbH, Switzerland
http://www.jekejeke.ch/idatab/doclet/prod/en/docs/05_run/package.html

[3] Android 2.3.3 Platform, Google Inc., July 2011
http://developer.android.com/sdk/android-2.3.3.html

[4] Java 2 Platform Standard Edition 5.0, Tiger, Sun Microsystems, 2004
http://www.oracle.com/technetwork/java/javase/index-jsp-135232.html

[5] Android 1.6 Platform, Donut, Google Inc., September 2009
http://developer.android.com/sdk/android-1.6.html

[6] Android 2.2 Platform, Froyo, Google Inc., May 2010
http://developer.android.com/sdk/android-2.2.html