
Developer's Reference

NetPhantom Eclipse

Version 7.7

Document Revision 2

3 October 2024



Mindus SARL

NetPhantom®

Version 7.7

© Copyright Mindus SARL, 2024. All rights reserved.

Information in this document is subject to change without notice. Companies, names, and data used in examples are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Mindus.

Mindus may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. The furnishing of this document does not give you the license to these patents, trademarks, copyrights, or other intellectual property rights except as expressly provided in any written license agreement from Mindus.

Phantom® and NetPhantom® are registered trademarks of Mindus. Java is a trademark of Sun Microsystems Incorporated. ActiveX, Microsoft, Windows are either registered trademarks or trademarks of Microsoft in the United States and/or other countries. IBM is a registered trademark of International Business Machines Corporation. Other products and company names mentioned herein may be the trademarks of their respective owners.

Mindus SARL

**1 Rue du Gabian
MC-98000 Monaco
MONACO**

Telephone: +377 99 90 32 66
Web: <https://netphantom.com>
E-mail: info@netphantom.com

Support

Phone: +377 99 90 32 66
E-mail: support@netphantom.com

Contents

1	Introduction	3
1.1	Overview	3
1.2	System Requirements	3
1.3	NetPhantom Quick-Start Installation.....	4
1.4	Preferences for NetPhantom and Eclipse	5
1.5	REXX Editor.....	6
	Syntax coloring	6
	Content assist.....	6
	Outline.....	7
	Hyperlinks	7
	Procedure, label and comment folding	7
	Occurrences marking.....	7
	Matching Do/End keyword and parenthesis matching selection.....	7
	Spell check	7
	Refactoring.....	7
	Preferences	8
1.6	Editor settings for a better Eclipse experience	8
1.7	Upgrade Editor and Server	9
2	Getting Started – NetPhantom Samples	11
2.1	Opening the NetPhantom Samples in a Workspace	11
2.2	NetPhantom Toolbar	12
	Drop-down menu.....	12
2.3	NetPhantom Perspective.....	12
2.4	Project Explorer.....	13
2.5	Views Contribution	13
2.6	Digest View	14
	Tree mode.....	14
	Digest mode.....	14
	Unresolved mode.....	14
2.7	References View.....	15
	Inbound References elements.....	15
2.8	Explorer Elements	16
	File-based Elements	16
	Control Elements.....	16
	Tool- and Statusbar Control Elements.....	17
	Other Non-file Based Elements.....	17
2.9	Refactoring	17
3	NetPhantom Development	19
3.1	Project Directory Structure.....	19
3.2	Creating NetPhantom Project in Eclipse	19
3.3	Creating NetPhantom Project in the Editor	21
3.4	Importing Existing Projects into the Editor	22
	REXX and Phantom Host Macro Conversion	24
	File Conversion – OEM Code Page	24
	Specialized Text File Code Page	24
3.5	Importing Current Project in Eclipse Workspace	24
3.6	Compile distribution.....	25
	Remove Say statements in Compiled REXX	25
3.7	Runtime application	25
3.8	Automatic Client Jar.....	26
	Configuring Automatic Client Jar	26
4	Using NetPhantom and Eclipse	27
4.1	NetPhantom and Eclipse coexistence	27
4.2	REXX Compilation in Eclipse	27
4.3	NetPhantom Console.....	29

4.4	File structure	29
	Description of all the entities in the file structure	30
4.5	NetPhantom Editors in Eclipse	31
4.6	Open Object Source from Editor	31
4.7	Filter function	31
	Other NetPhantom filters	32
	Configuration	33
	Project Explorer	33
	Package Explorer	34
4.8	Application references	34
4.9	Eclipse and Java settings for NetPhantom projects	35
4.10	Eclipse conversion of Java Project to NetPhantom	35
4.11	Creating a NetPhantom Object in Eclipse	35
	NetPhantom REXX Object	36
	NetPhantom Java Object	37
5	Application Execution	39
5.1	Debugging the Server application	39
6	Troubleshooting	41
6.1	Problems using 64-bit Eclipse with NetPhantom	41
6.2	NetPhantom menus, items or views do not appear	42
6.3	The NetPhantom Project layout is wrong	42
6.4	The NetPhantom project is inconsistent	43
6.5	Connection between Eclipse and NetPhantom is disrupted	43
6.6	The Java code is compiled with errors or warnings	44
6.7	Load error “Unsupported ClassVersionError” in application	44
6.8	The NetPhantom Project is marked with a warning	45
6.9	Warning “bootstrap class path not set...”	46
6.10	Error “Panel MODAL disposed...”	46
6.11	Error “The system is out of resources”	47
	Eclipse	47
	NetPhantom Editor	47
	NetPhantom Application	48

Preface – Intended Audience

The *Developer's Reference to NetPhantom Eclipse* is not intended as end user documentation but rather as a guide and reference for NetPhantom application developers. It should be noted that the documentation has been written with the assumption that the reader is familiar to the Java programming language as well as the Eclipse IDE (Integrated Development Environment).

The convention in this document is that only functions specific to the NetPhantom – Eclipse integration is handled here. The basic NetPhantom functionality is described in the *NetPhantom User's Guide* document.

1 Introduction

The NetPhantom Editor and Eclipse IDE have been integrated. This integration means that a NetPhantom application can be developed using the Editor for the NetPhantom specific parts and the Eclipse environment for handling the developed source code, whether written in REXX or Java.

The integration between NetPhantom and Eclipse is accomplished with NetPhantom plug-ins that are installed in Eclipse.

1.1 Overview

Eclipse is the de-facto standard IDE for software development. With the integration, the NetPhantom developer gets access to large number of tools:

- Editors with help and assistance, error checking, code completion, annotations, code formatting, refactoring, etc.,
- Debugger,
- Profiling tools,
- Structured editors and browsers.

The NetPhantom/Eclipse development environment consists of four major parts:



Terminal Editor,



Panel Editor,



Eclipse,



NetPhantom Server; used for running test/debug versions of the application(s) being developed.

When working with the development environment, the icons above switch between the parts.

1.2 System Requirements

NetPhantom Editor requires 64-bit Windows to run, thus the Operating System must be Windows. The Editor process is a 32-bit process that requires a 32-bit version of the Java Development Kit (containing a Java Compiler), i.e. Java 1.8 JDK update 151 or better.

- Eclipse 2019-12 in 32-bit or 64-bit version, or the one we supply, which is in 64-bit and much later version, currently Eclipse 2024-09 version 4.33,
- Java SE Development Kit 8 update 151 or better, in 32-bit version, but Java version 21 or better is recommended. All versions of Eclipse 4.23 or better require at least Java 11 to run, in 64-bit.
- Windows 7 to 11.

Note: it is the *Java SE Development Kit* that is required, not the *Java SE Runtime Environment*.

1.3 NetPhantom Quick-Start Installation

The NetPhantom Quick-Start installation is the fastest way of getting up and running. This installation comprises of:

- NetPhantom Server and Editor version 7.7.0 (7.70).
- OpenJDK Development Kit version 17 Update 11 (32-bit).
- OpenJDK Runtime Environment version 8 update 422 (64-bit).
- OpenJDK Runtime Environment version 21 Update 4 (Eclipse JustJ, 64-bit).
- Eclipse 2024-09 version 4.33 for Java Developers (64-bit).

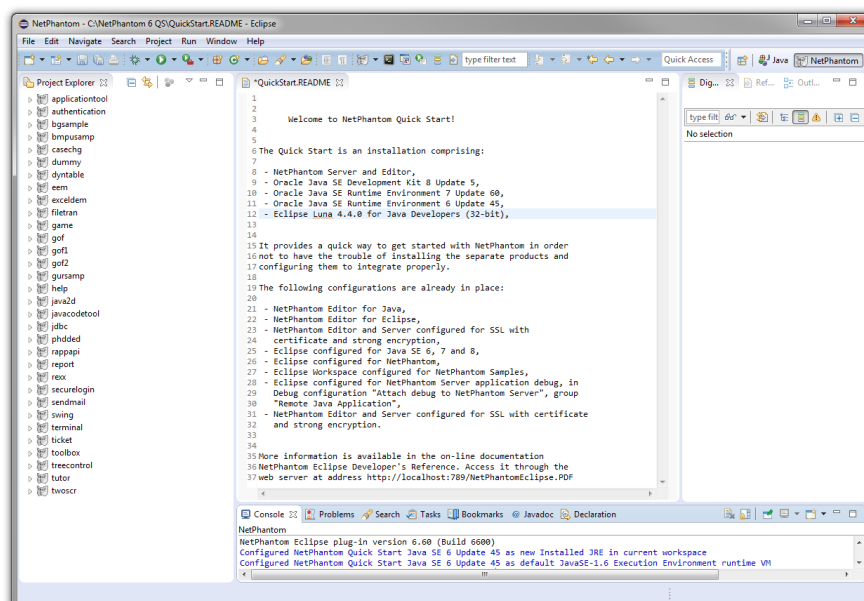
It provides a quick way to get started with NetPhantom in order not to have the trouble of installing the separate products and configuring them to integrate properly.

The following configurations are already in place:

- NetPhantom Editor for Java.
- NetPhantom Editor for Eclipse.
- Eclipse configured for Java SE 1.8, Java 9 to 23 Runtime Environments.
- Eclipse configured for NetPhantom.
- Eclipse Workspace configured for NetPhantom Samples.
- Eclipse configured for NetPhantom Server application debug, in Debug configuration *Attach debug to NetPhantom Server*, group *Remote Java Application*.
- NetPhantom Editor and Server configured for SSL with certificate and strong encryption.

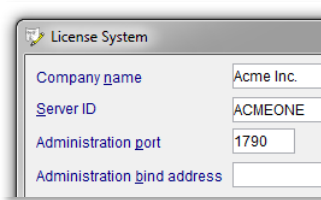
The installation is done in the following way:

1. Run the installer `NetPhantomQuickStartSetup.exe`.
2. At the completion of the installation wizard, leave the option **Launch NetPhantom Quick Start** selected and press **Finish**. This will start NetPhantom Editor.
3. Complete the **Welcome** dialog box, and you will end up with a complete installation, up and running, with the Samples Workspace looking like:

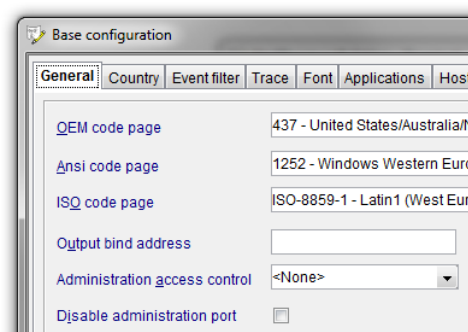


1.4 Preferences for NetPhantom and Eclipse

NetPhantom and Eclipse communicate over the NetPhantom Administration Socket. To view the settings, select the menu item **Options – License** in the Editor:

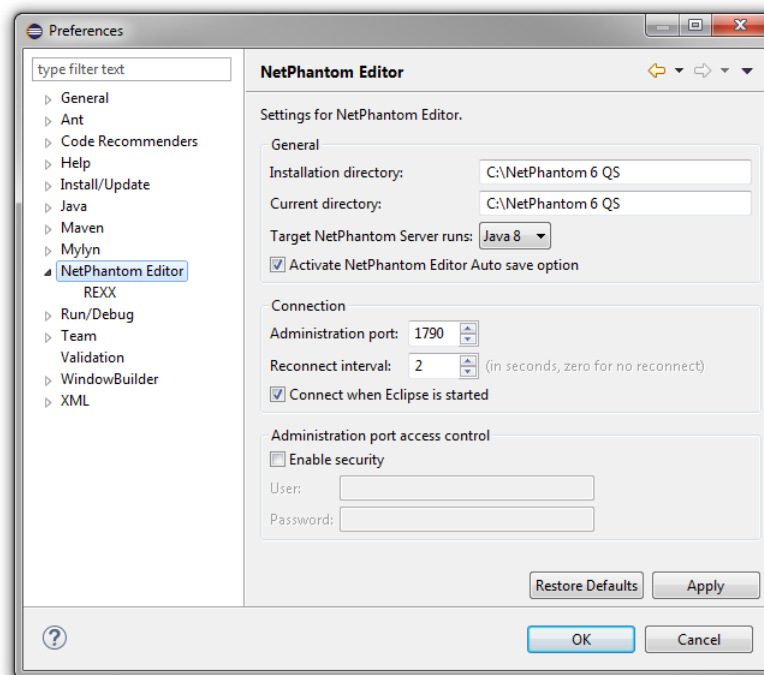


The default port is **1790** and the **Administration access control** is normally not defined. To view the settings for authentication, select menu item **Server – Base configuration**:



The combobox for the **Administration access control** specifies the Access Control that is allowed to access the port. *The checkbox **Disable administration port** should not be selected as this will make communication with Eclipse impossible.*

The Eclipse dialog box for NetPhantom Preferences is shown by the Eclipse menu item **Windows – Preferences**, followed by selection of **NetPhantom Editor** in the tree on the left-hand side:



The **Installation directory** along with the **Current directory** is normally managed by NetPhantom Editor.

The option **Target NetPhantom Server runs Java 8** is a state that is automatically set from the NetPhantom Editor when the connection is established or when the Editor changes the target Java version for the server in the Java Runtime Environment dialog box. This setting is used to check the Java compliance level when creating new NetPhantom projects.

The option that is turned on by default is **Activate NetPhantom Editor Auto save option**. This option causes NetPhantom Editor to automatically save all files (panels, screens, application, etc), when it is required. This is very important as Eclipse causes NetPhantom Editor to switch between NetPhantom projects or applications.

The **Connection** to the Editor is also defined with the **Administration port** that normally is 1790, and the other options are used for a reconnection timer or reconnection to the Editor when Eclipse is started.

If an *Access Control* is selected in the Editor and requires *User Authentication*, the Eclipse NetPhantom Settings provides the secure **User** and **Password** entry fields. The password is saved in a safe encrypted Eclipse preference store.

1.5 REXX Editor

Source files with `.rex` file extensions use the NetPhantom REXX Editor in Eclipse. This editor supports the following features:

- syntax coloring,
- source outline,
- procedure, label and comment folding,
- occurrences marking,
- matching *Do/End* keyword and parenthesis matching selection,
- spell checking,
- rename and refactoring.

Syntax coloring

The REXX source is syntax colored for the elements:

- comments,
- strings,
- numbers,
- REXX keywords,
- symbols,
- variables (global, private and stem variables),
- procedures and label sections,
- REXX built-in and NetPhantom functions.

Content assist

Content assist is available for REXX keywords, REXX built-in and NetPhantom functions. It is activated using *Ctrl+Space* and will complete the word under the cursor, with parameters in case of a function. Functions are also completed when the left parenthesis key '(' is pressed. A pop-up listbox is presented for selection of content assist in case of an ambiguous word.

Outline

The Eclipse outline view shows the REXX source outline with the elements:

- procedures,
- label sections,
- variables (global, procedure private in combination with stem variables or arrays).

Selections in the outline view will annotate all occurrences of the word in question in the editor and moving between annotations can be done using the *Next* or *Previous annotation* buttons (*Ctrl+*, and *Ctrl+*,).

When double-clicking a word in the editor, the word is selected in the outline view. When double-clicking the word in the outline, the cursor is moved to the declaration of the word, e.g. the procedure, label or variable declaration.

Hyperlinks

Procedures, label sections and variables are “hyperlinked”, i.e. if pressing the *Ctrl* key and clicking on the word in question, the cursor is moved to the definition of the word, e.g. the procedure, label or variable declaration.

Procedure, label and comment folding

Procedures, label sections and comments are foldable using the + and – icons in the left margin.

Occurrences marking

The occurrences marking select all occurrences of a word under the cursor after half a second of non-editing of the REXX source. Words marked are variables, functions, procedures, labels, and strings. The words are marked once the cursor has not been moved for half a second.

Matching Do/End keyword and parenthesis matching selection

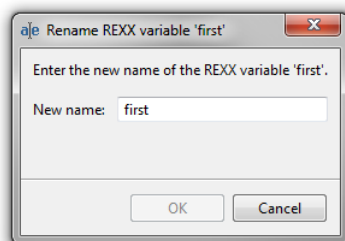
When the cursor is placed before or after the *Do* or *End* keyword, the matching *End* or *Do* keyword is selected. The cursor may be moved to the matching location using the *Next* or *Previous annotation* buttons (*Ctrl+*, and *Ctrl+*,). This applies to the *Do/End* keywords.

Spell check

Spell checking is performed in *Comments* and *Strings* by default and is configurable. The spell check language and dictionary are configured using standard Eclipse preferences.

Refactoring

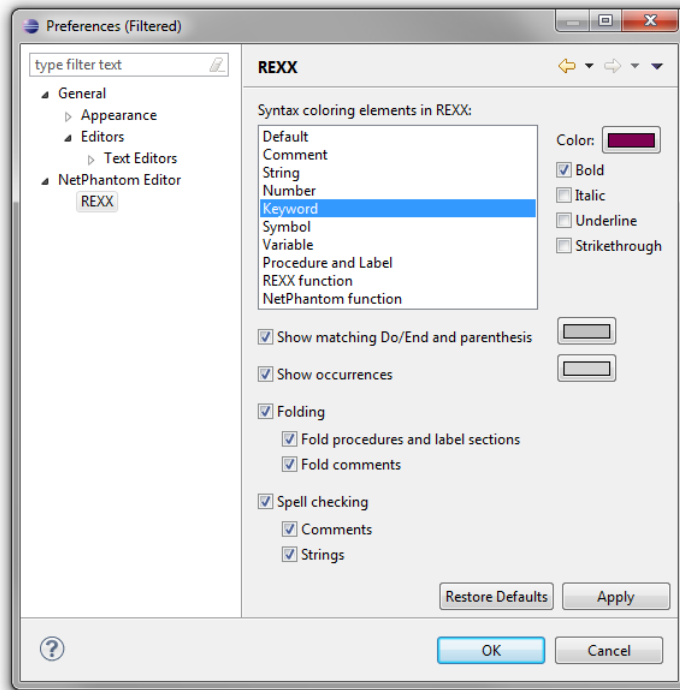
Variables, procedures and labels can be renamed globally, also called *Rename refactoring*. When the cursor is on a word of a variable, procedure or label, select the menu item **Edit – Rename *nn*** where *nn* is replaced with the word description, e.g. *REXX variable 'first'*. Refactoring is also available from the *Outline view* using the context *pop-up menu*. The rename dialog box is displayed and you fill in the new name:



Note: A refactoring operation can be undone and redone as other editor operations; however, a message box may be displayed stating the operation will affect other files apart of the one being edited. This is due to background compilation process to NetRexx, then to Java.

Preferences

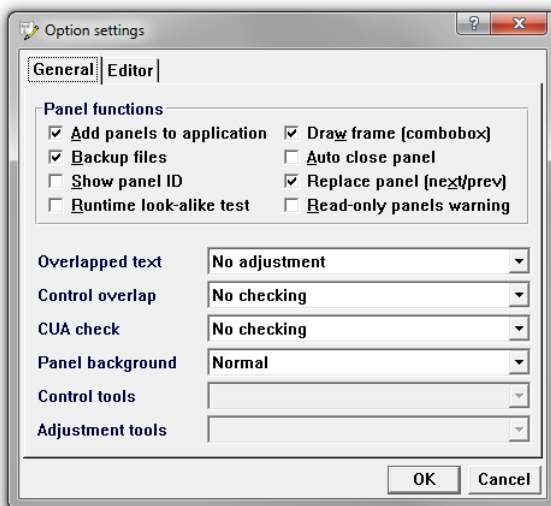
The REXX preferences are configured in Eclipse using the **Window – Preferences** menu item or with the REXX editor pop-up menu using **Preferences**:



1.6 Editor settings for a better Eclipse experience

It is suggested to change settings in NetPhantom Editor to smooth the integration with Eclipse, typically when switching projects.

To do so, select the menu item **Options – Settings** in the Panel Part.



The dialog box shows the most aggressive settings to avoid most message boxes.

The settings that you should consider changing are the ones that inhibit questions in the form of message boxes. This will interrupt the operation that Eclipse initiated until they are answered.

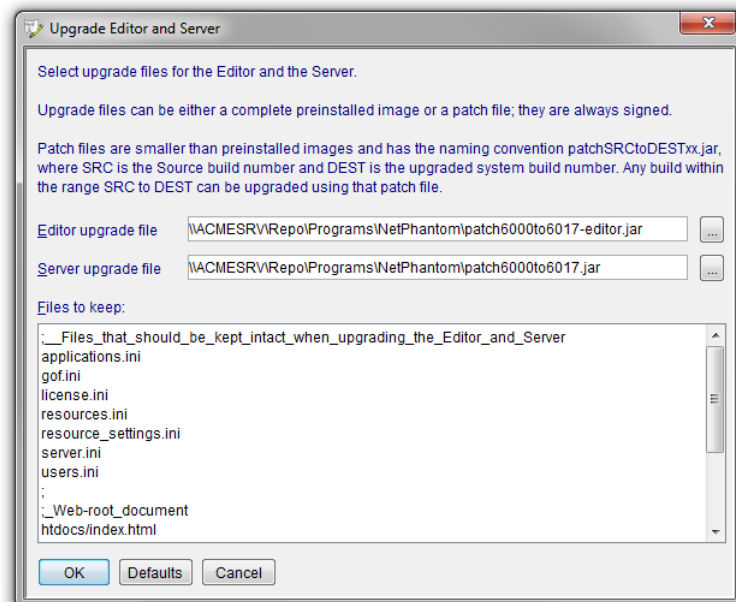
1.7 Upgrade Editor and Server

As the Editor is integrated with the Server, it requires upgrades of both parts. From NetPhantom version 6, Eclipse is integrated with the NetPhantom Plug-ins for Eclipse, and the plug-ins also requires upgrading to match the Editor. The upgrade facility makes this operation simple while retaining the complex configurations that may have been done.

Upgrades of the Server can be performed using a preinstalled image named `preinstall.jar` or `preinstallNNNN.jar` where *NNNN* stands for the Build number. Smaller upgrade files are also available as patch files named `patchSRCtoDEST.jar`, where *SRC* is the Build number from where the upgrade applies and *DEST* is the final Build number after applying the patch. E.g. `patch7000to7485.jar` upgrades all builds 7000 to 7484 up to 7485.

The upgrade facility uses two files for the upgrade, one for the Editor and one for the Server. The Editor's file has the same base name as the Server's, with `-editor` appended before the `.jar` file extension.

To upgrade the Editor and Server at the same time, select the Editor menu item **Help – Upgrade Editor and Server**. The following dialog box is shown:



Fill in the two file names, the first for the **Editor upgrade file**, e.g. `patch7000to7485-editor.jar`. If you browse for the file, the **Server upgrade file** name is filled in if the file exists.

The configuration files saved in the upgrade are specified in **Files to keep**. One file should be specified per line and can contain the Windows `*` `?` `^` wild cards. Specifying a sub-directory in front of the file name is also supported, as `subdir/file` (*not* `/subdir/file`). Use the forward slash (`/`) and not backslash (`\`). E.g. `htdocs/myapp/*` will retain all files in the `htdocs/myapp` directory.

Exit Eclipse if it is running, as the upgrade operation also will upgrade the NetPhantom Plugins for Eclipse when the Editor is restarted.

2 Getting Started – NetPhantom Samples

This section describes how you get started, with the NetPhantom Samples as Eclipse projects.

The Samples provides good examples of the new NetPhantom functions available. As you work more with Eclipse and NetPhantom you will find that there are many short-cuts or fast paths to function or ways to find what you are looking for.

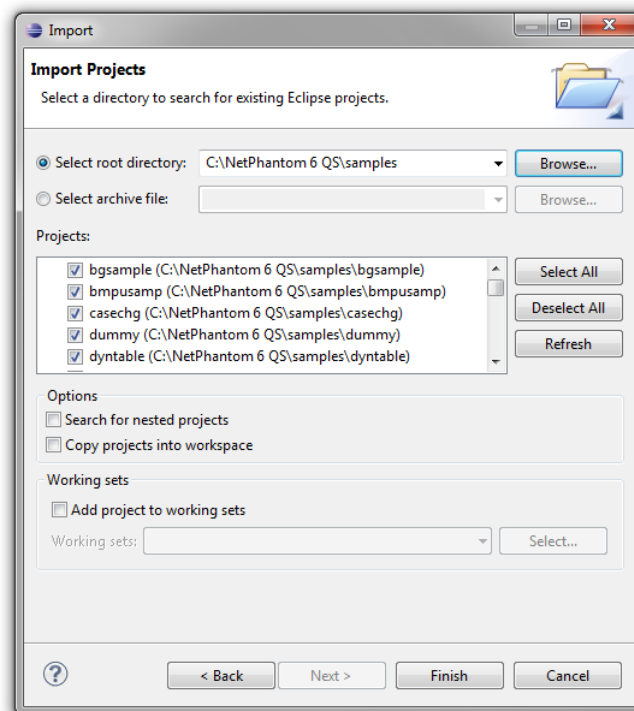
There are several samples provided with NetPhantom in separate projects, all opened in the same Eclipse Workspace. The Editor will switch between the samples automatically as required.

2.1 Opening NetPhantom Samples in a Workspace

If the NetPhantom Quick-Start installation is installed, you can open the Eclipse Samples Workspace with the short-cut **Eclipse - Samples Workspace**. This short-cut can also be used for a normal installation if the Eclipse configuration has been done in the Editor with the menu item **Options – Configure Java environment**.

To open the samples manually, follow the procedure described here to open the NetPhantom Samples provided with the Editor in an Eclipse workspace.

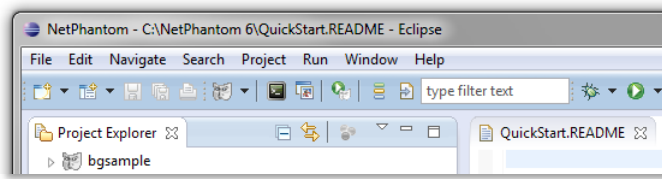
From Eclipse, select the menu item **File – Import**, then select item **Existing Projects into Workspace**.









Enter `C:\NetPhantom 7\samples` (or the path to the samples if installed in another location) in the **Select root directory** and press the *Tab* key to populate the **Projects** list. Select the samples to import and press **Finish**.

2.2 NetPhantom Toolbar

The Eclipse workspace, when using NetPhantom, has the NetPhantom toolbar:

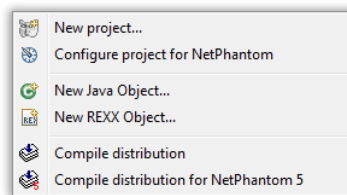


The toolbar has the following icons and functions:

-  Use the arrow to display a drop-down menu as described below, or the button action to connect to or disconnect from NetPhantom Editor.
-  Quick access button that brings the Terminal Editor window to front,
-  Quick access button that brings the Panel Editor window to front,
-  Launches the NetPhantom Client for the selected project from Project Explorer.
-  Open the Digest view with the selection in Project Explorer.
-  Opens the Reference view with the selection in Project Explorer.

The last item on the toolbar is the filter entry field, used to filter displayed elements in the Project Explorer for easy access. Filtering is very powerful and is described in detail in *Chapter 4.7 Filter function*.

Drop-down menu

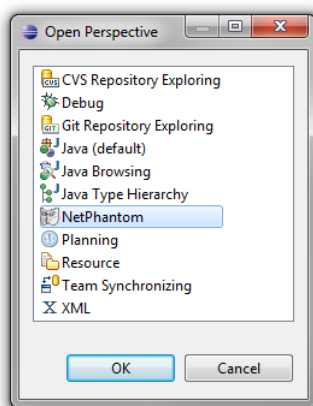


Use these menu items to create a new NetPhantom project, Objects or to compile a distribution Jar file of an application.

2.3 NetPhantom Perspective

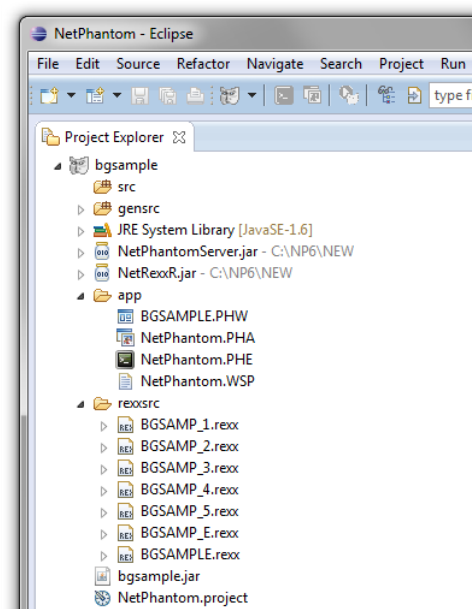
The NetPhantom Perspective is optimized for NetPhantom in terms of open views, positioning, toolbars, and menu items.

To open it, select the menu item **Open – Perspective – Other** and select **NetPhantom**.

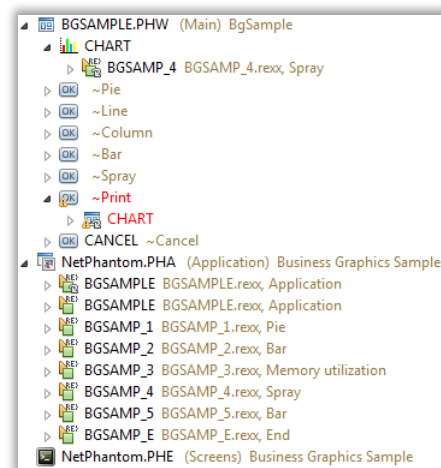


2.4 Project Explorer

A NetPhantom project in the Project Explorer looks like the image below when a connection to NetPhantom Editor is not present. If you look carefully, you will notice that the information provided in the first image (see BGSAMPLE item in particular) is not as detailed as the second image (that shows just the BGSAMPLE item).



If Eclipse is running without NetPhantom Editor, it only displays the resource contents, i.e. the folders and files. If a connection to the Editor exists, the project files will be enhanced with structural information depending on the file type.



The details can be expanded, and what is displayed depends on the item type, its connections in the application, references, etc. As you note above, unresolved errors are shown as well as warnings, propagating upwards in the tree, here shown in red with a small warning tag.

2.5 Views Contribution

NetPhantom contributes to Eclipse to in the Project Explorer as shown above. Two special NetPhantom Views show information relating to the selection in the Project Explorer.

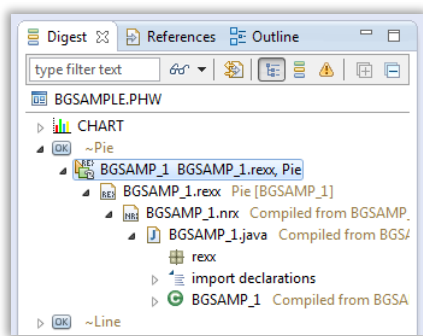
The views can display the current selection in Project Explorer or stay *Linked*. When activated, the view will update its contents regarding the new selection.

The views have active filtering that is used to select elements containing only the entered filter text. This text is entered instead of *filter text* in the entry field at the top of the view.

A view pull-down menu provides options for sorting **alphabetically** and by **type**, as well as to include Java elements in filtering. As Java elements can be expanded into very large and deep trees, this can require heavy processing, so be careful using this option as Eclipse may seem to be unresponsive. Due to this fact, the **Expand all** toolbar button is disabled when in **Tree** mode.

2.6 Digest View

Located by default to the right in Eclipse, the **Digest** view presents a digest of a selection in Explorer. The Digest View shows three types of information for a selection in the Project Explorer, also called **Tree**, **Digest** and **Unresolved** modes.

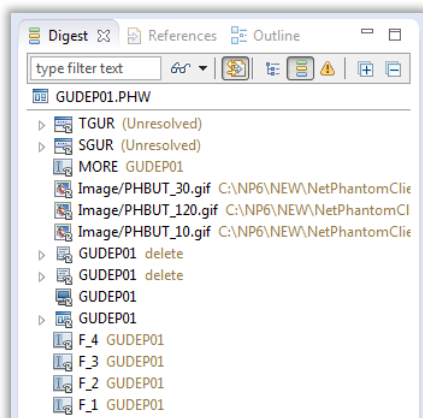


Tree mode

Looking a lot like Project Explorer, the mode provides optional expansion of the elements. Use the **View** menu to specify the initial expansion level of the tree when the view input is changed with a new selection.

Digest mode

The element and all its children's references are shown in a "flat" digest structure. Most of the elements shown can be expanded showing further information.



Unresolved mode









This mode is very useful when working with problems in an application.

2.7 References View

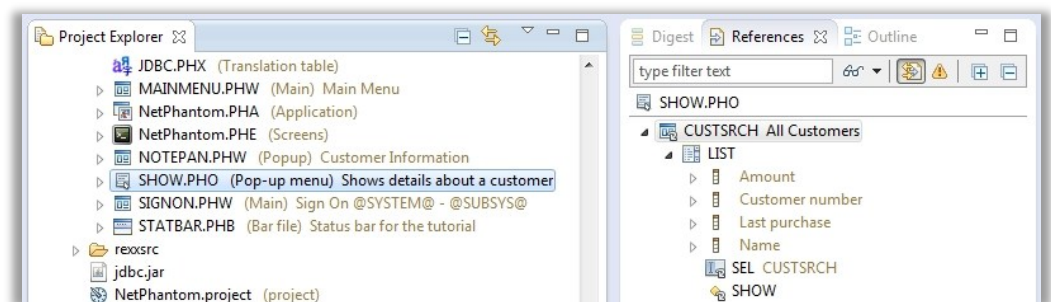
Much like the Digest view, the **References** view shows information about the selected element in the Project Explorer. This view shows all *inbound references* to the selected element. E.g. if you select a REXX Object, all panels and controls that refer to the Object are shown.

Inbound References elements

The table below shows the inbound references that can be shown in the References view. When controls are shown as referencing elements, this also includes toolbar and status bars.

-  The screen will show all elements that are connected to the screen. The elements are: panels, controls, menu items, pop-up menu items and text prompts that use a @*REF@ reference to the host field content.
-  The host field will show the referencing elements: panel (when host field is referenced from the title text), controls, menu items, pop-up menu items, message boxes. Referencing elements not directly linked to the screen of the host field will not appear, e.g. if the panel has no host screen but attaches to the current screen when shown.
-  Objects, the REXX or Java Object, the source file, or the generated source files (NetRexx and Java). This element will show connections to the Object: application, panels, controls, message boxes.
-  Panels show the controls, menu items and pop-up menu items that refer to the panel using **Next panel**. Notebooks are also shown as they refer to the panel as the notebook page.
-  Controls (the actual image to the left depends on the selected control, here a list box) show connections from other controls, menu items or pop-up menu items in the same panel. References to the controls are defined as e.g. the **Double-click ID**, or the **Panel control connection**.
-  Combobox files show the comboboxes and spin buttons that use them.
-  Bar template files for toolbar or status bars show the panels that use them.
- TEXT** Text IDs in text files (PHM) show the elements that refer to them in *Text prompts* as @*TEXTID@ or directly refer to the Text ID as e.g. a **Send string** with **String is Text ID**.
-  Images shows push button, checkboxes, radio buttons, toolbar buttons and external images that refer to them.

The layout of the References view shows the inbound references on the first level. All the referring items can then be expanded and will then show the (containment) structure as it is seen in the Project view.

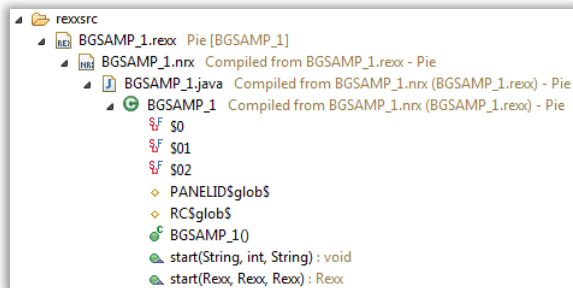


The example depicts the SHOW pop-up menu in the Project Explorer view on the left. The linked References view will then show the CUSTSRC panel which refers to the menu. Contained in that panel, the other components and references are shown.

This duplication of information is deliberate for the purpose of usability of the tool.











2.8 Explorer Elements




NetPhantom displays its elements in various places, mostly in the Project Explorer, Digest and References Views. An example of the `rexxsrc` folder showing a REXX source file with its nested children's elements:



File-based Elements













The file-based elements are added as children to files in most places a file is shown in Eclipse. This table describes the various elements and what their children can be.











-  The project file, no children. Opening this file will open the NetPhantom Project settings dialog box.
-  Host terminal data. The children are host screens.
-  Application data. Children can be: Objects, pop-up menus, panels and toolbar template files. If the application itself refers to an Object (from the application definition) and a Text ID (from the application title).
-  The panel can show most other elements as panels are complex in structure.
-  Pop-up menus contains submenus and pop-up menu items.
-  Combobox files show their line contents with possible host conversion using Text entry elements.
-  Bar template files shows its bar controls.
-  The text table file shows its text IDs elements.
-  REXX files shows the NetRexx generated source.
-  NetRexx files shows the Java generated source.

The elements   , i.e. translation table, help ID file and tooltip text file, do not provide children.

Control Elements











The panel Control elements show the references they have to host fields, Objects, text IDs, panels and other controls. What exactly is shown depends on the control type. The control types images are (including bar controls):

- | | | |
|---|---|--|
|  Static text |  Output text |  Entry field |
|  List box |  List box column |  Push button |
|  Checkbox |  Radio button |  Group box |
|  Rectangle |  Frame |  Business graphics |

	User window		Multi-line entry field		Combobox
	Spin button		Sub window		Notebook
	Notebook page		External image		Bar control
	Tree control				













Tool- and Statusbar Control Elements

The bar control has the following bar control elements:

	Bitmap		Date/Time		Host field
	Keyboard indicator		Message		Spacing
	Button		Standard button		Text
	User draw				

Other Non-file Based Elements

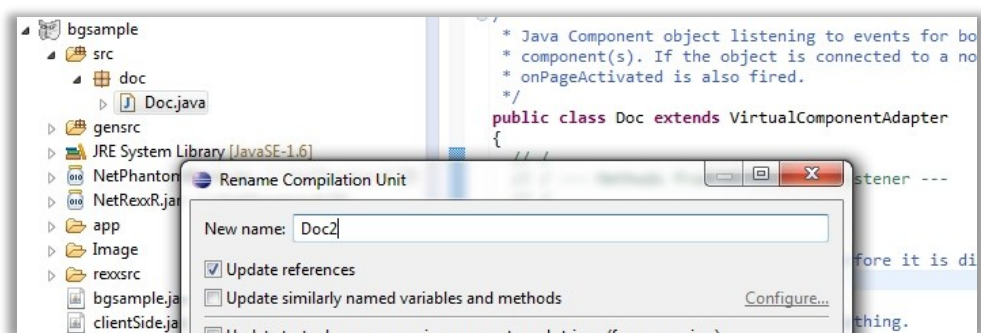
The table below shows other non-control elements that are not file based. The REXX and Java Objects icons belong to this group as they are the NetPhantom Object definitions, not the source code.

	REXX Object		Java Object		Screen
	Screen identification		Host field		Message box
	Menu bar		Pull-down menu		Menu item
	Hidden panel		Text ID		Text entry

2.9 Refactoring

The NetPhantom integration allows for refactoring between Eclipse and the NetPhantom editor. This is a powerful tool in the application development and maintenance phases.

In the example illustrated below, a Java class is renamed by selecting **File – Rename ...** to initiate the refactoring action.



What will happen is that the Java class will be given a new name in the source file. Optionally, references to this class in other Java files will be updated. So far, this is standard Eclipse JDT functionality.

In parallel with the Eclipse refactoring, the NetPhantom integration will update any object definitions which reference the Java class in question. Please note that for this mechanism to work, the NetPhantom Editor must be running and connected to the Eclipse instance responsible of initiating the refactoring.

3 NetPhantom Development

The application which is under development is called a *Project* and is a *directory*. Is is opened as a project in both the Editor and Eclipse.

Previous versions of NetPhantom used a *Terminal File* .PHE and an *Application File* .PHA that could be located in different places and have different names. With NetPhantom Eclipse integration, this has been converted into this new *Project Directory Structure*.

3.1 Project Directory Structure

The project directory has the following structure in NetPhantom Editor:

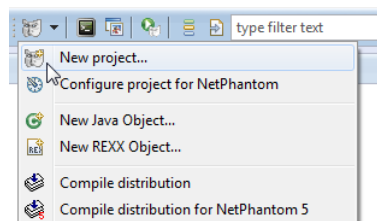
Directory	Description
app	Application files such as screens, panels, pop-up menus, tables (text table, translation, etc).
rexxsrc	REXX Source directory used for .rexx files. If there is a compilation problems, .rexxerr files are placed here, with the same base file name as the compiled REXX file.
src	Java Source directory, used for .java files.
gensrc	Generated sources files from compiled REXX, .nrx for NetRexx, .java for compiled NetRexx (used to set breakpoints in the NetPhantom Server application in compiled REXX code).
bin	Directory where compiled Java class files are placed as well as compiled REXX sources in turn compiled into Java class files.

REXX files are compiled into NetRexx in the package rexx. Thus a subdirectory rexx may be present under gensrc and under bin.

The NetPhantom project contains the file NetPhantom.project in its root, and in the app directory the files NetPhantom.PHE and NetPhantom.PHA for the *Terminal Screens* and the *Application* respectively.

3.2 Creating NetPhantom Project in Eclipse

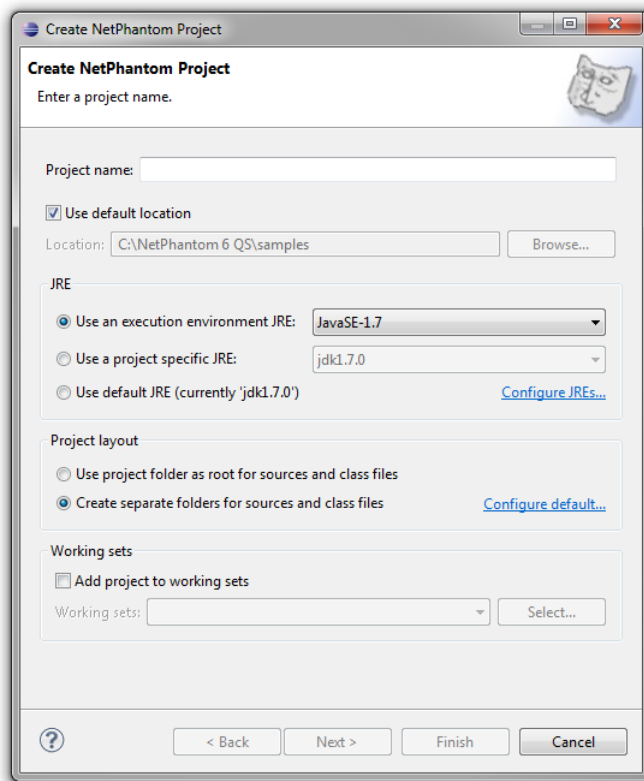
The NetPhantom project wizard is shown in Eclipse from the NetPhantom menu shown below.



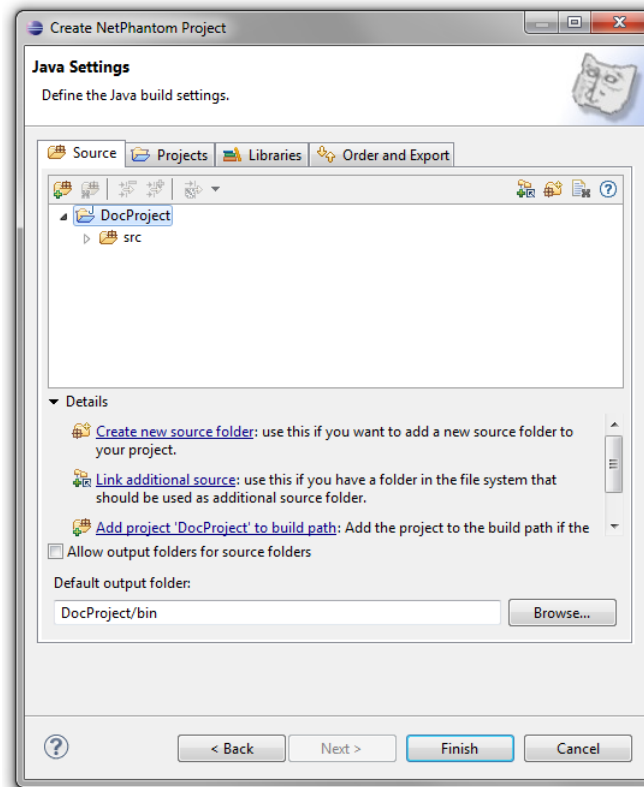
This will start an Eclipse wizard which looks similar to that of a Java project.

Enter the **Project name**. This will also become the directory name. For normal use, do not change the option **Use default location**.

Select the **JRE** with the option **Use an execution environment JRE** as *JavaSE-1.8* or *JavaSE-9* or better depending on the target NetPhantom Server version. You may choose *JavaSE-1.8* even if the Server is using Java 11 or 8, only the created project is affected.



The **Project layout** must use the default option **Create separate folders for sources and class files**.

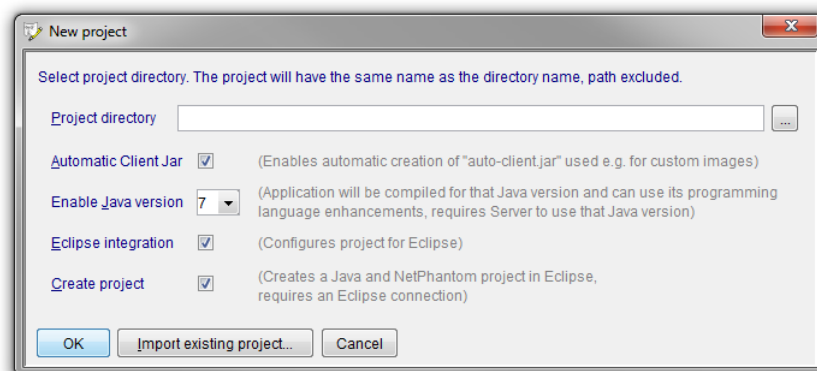


The second page in the wizard enables selection of the **Source folder** and the **Default output folder**. Leave them as default, i.e. `src` and `bin` respectively to be built correctly in the Editor.

When the wizard completes, additional NetPhantom files and directories are created, i.e. the directories `app` and `gensrc\rexx`, the files `NetPhantom.project`, `app\NetPhantom.PHE` and `app\NetPhantom.PHA`.

3.3 Creating NetPhantom Project in the Editor

A NetPhantom project can also be created using the NetPhantom Editor menu item **File – New project**:



Fill in the **Project directory** that also is the project name in Eclipse and select the options for Eclipse.

Select the option for **Automatic Client Jar** if you wish to use a Client Jar file for e.g. images and have the folder *Image* created in the project.

If the Target NetPhantom Server is configured to run Java 8 or 11 in the Java Runtime Environment configuration, select the **Enable Java version** option to enable programming with Java 8 or 11 compliance.

The **Eclipse integration** must be selected to use Eclipse. The **Create project** option can only be available when a connection to Eclipse exists and the project is selected to use Eclipse integration. It creates the necessary structures for the project to be visible in current Eclipse workspace.

Select the push button **Import existing project** to import applications from previous versions of NetPhantom or Phantom Hurricane.

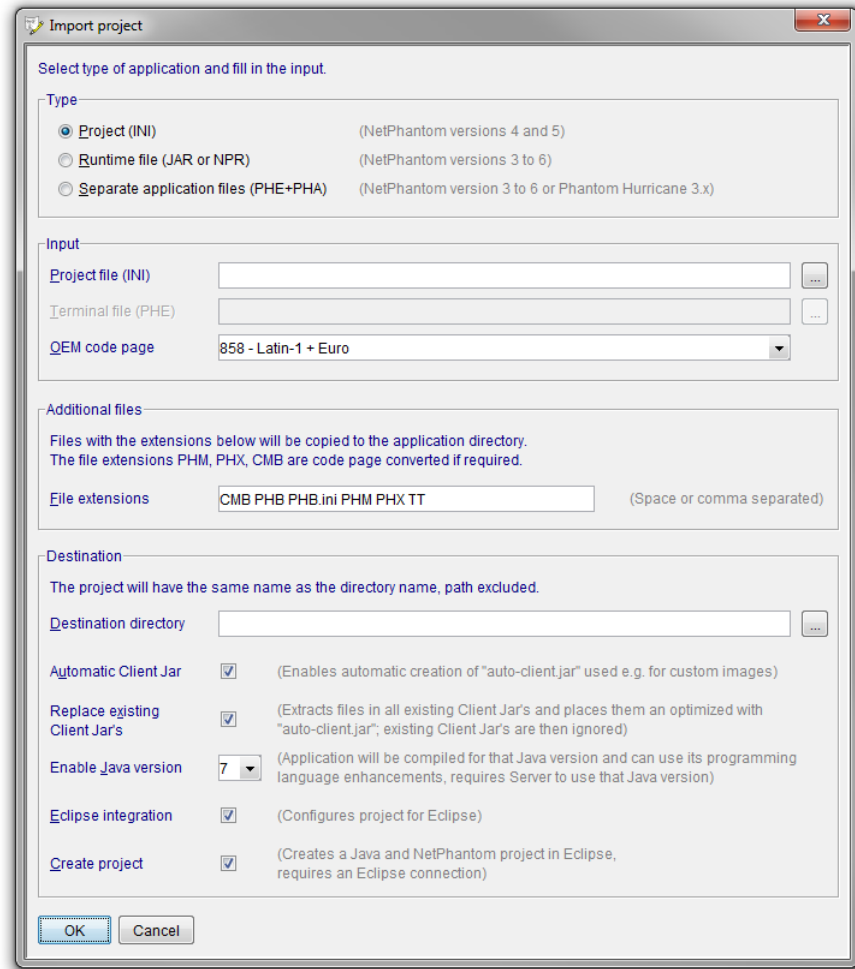
3.4 Importing Existing Projects into the Editor

Previously developed NetPhantom projects, applications or runtime files can be imported into the new projects for NetPhantom 7.

Import can be done from the following types where the upper-case texts below indicate the file extensions:

- **JAR**: NetPhantom 5, 6 and 7 Jar files used as Server applications (but not a NetPhantom 5 Jar that has been compiled with NetPhantom 6 or better),
- **NPR**: NetPhantom 3 runtime applications running in a Server,
- **INI**: NetPhantom 5 projects (not from an unpacked Jar produced by compiling a NetPhantom 6 or 7 project for NetPhantom 5),
- **PHE+PHA**: Separate Terminal and Application files, all versions, including Phantom Hurricane files.

Use the Editor menu item **File – Import existing projects**:



Select **Type** of application or project files to import and the **Input** files depending on the type.

NetPhantom 6 and 7 projects use the Windows character set for textual files (text table, translation table, tooltip files, combobox files, and REXX source files) as opposed to previous versions using the OEM code page. It is therefore important to select the **OEM code page** that matches the input files.

Additional files that should be imported into the project are specified as **File extensions**. All files in the directory of the INI, JAR/NPR or PHA file that matches the file extension list will be imported. *The additional files are also code page converted where appropriate.*

Select **Destination directory** and if you wish to have the Eclipse integration configured directly and also to create the required Eclipse project files (`.project` and `.classpath`).

Select the option for **Automatic Client Jar** if you wish to use a Client Jar file for e.g. images and have the folder *Image* created in the project. The option **Replace existing Client Jar's** is used to extract all contents of possibly existing Client Jar files and place them in their respective directories. If a Client Jar file contains files in its root, they will be placed in a directory called *ImageRoot*. The previous Client Jar files will no longer be used in the project and are not included in the project.

If the Target NetPhantom Server is configured to run Java 8 or 11 in the Java Runtime Environment configuration, select the **Enable Java version** option to enable programming with Java 8 or Java 11 compliance.

The **Eclipse integration** must be selected to use Eclipse. The **Create project** option can only be available when a connection to Eclipse exists and the project is selected to use Eclipse integration. It creates the necessary structures for the project to be visible in the current Eclipse workspace. When both options are selected and the project import completes, you may be asked if you wish to import the project into the current Eclipse workspace, if Eclipse is currently running and connected to the Editor.

REXX and Phantom Host Macro Conversion

During the import, potential Phantom Macros (or host dialogs) are converted into REXX code. The REXX source files are also renamed to have the file extension `.rexx` and the base file name are set to upper case.

File Conversion – OEM Code Page

As previously mentioned, it is important to select the **OEM code page** correctly matching the imported application files.

The OEM code page can be retrieved from the NetPhantom Editor or Server settings that hosted the files, use the **Server – Base configuration** to view the setting.

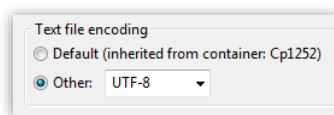
If it is a Phantom Hurricane file, you can display the code page by starting a Windows Command Prompt `cmd.exe` and enter the command `chcp`.

The files in NetPhantom 6 below are code page converted into the Windows (Ansi) code page, making it possible to edit files directly (e.g. in Eclipse) without code page conversion:

- Text table (`.PHM`),
- Translation table (`.PHX`),
- Tooltip text (`.TT`),
- Help ID file (`.PHH`),
- Combobox files (`.CMB`),
- REXX source files, typically with file extension `.CMD`.

Specialized Text File Code Page

NetPhantom supports using special code pages for text files. The first line in the file is `;#CODEPAGE=nnn` where *nnn* is the code page in question, e.g. UTF-8. UTF-8 can be used to enter double-byte characters such as Chinese. When using these specialized code pages, the Text Editor function in the Panel Editor should not be used. The Eclipse text editor should be used. It may also require configuration from its default. To change the Eclipse code page for a file, select it in the Explorer, right-click and choose **Properties**. Then change the code page setting as the example shows:



3.5 Importing Current Project in Eclipse Workspace

When the Editor has an open project, this project can be imported into the current Eclipse workspace, if Eclipse is running and connected to the Editor.

Select menu item **File – Import current project in Eclipse workspace**.

3.6 Compile distribution

A NetPhantom application distribution is a Jar file which contains everything necessary for running the application in a NetPhantom Server. The Compiled distribution Jar can also be used when building new applications, by referencing and using resources in other applications. How this is done is described in the *NetPhantom Administrator's and Developer's Guide* document.

Prior to NetPhantom 6, the **Compile distribution** operation created a Jar file of the compiled REXX Object sources.

The Compile distribution operation now compiles Java code as well as REXX code in a temporary directory. The produced Jar file is then extensively tested for referential integrity and correctness, using the NetPhantom 7 *Application references* operation.

All sources are recompiled with the compiler options as specified in the **Project configuration**; for REXX using the option **Remove Say statements in Compiled REXX**, and for Java using the **Debug information** option and the **Additional options** specified on the **Java** page.

Please note that an application which contains errors will not be possible to compile into a distribution. It is however possible to (test-) run an application in the Editor.

Remove Say statements in Compiled REXX

Using the **Remove Say statements in Compiled REXX** option will change how the REXX code is interpreted; any **Say** statements will be disregarded in the compilation. This subsequently means that syntax errors may occur as in this example:

```
DO
  IF var1 > 0 THEN say 'var1:' var1
END
```

The syntax error is a dangling THEN statement that had an END. To write code which still works, the following construct can be used:

```
IF var1 > 0 THEN DO
  say 'var1:' var1
  NOP
END
```

Another example for changed behavior would be:

```
IF var1 > 0 THEN SAY 'var1:' var1
rc = HostSend('@E')
```

and would result in:

```
IF var1 > 0 THEN
  rc = HostSend('@E')
```

If code containing SAY statements is carefully crafted, the option **Remove Say statements in Compiled REXX** is powerful because it effectively eliminates “debug” output in an application.

3.7 Runtime application

The running NetPhantom application distribution can be dependent on external files in runtime (e.g combination box files). The default location of these files is a directory called `app` located in the same directory as the application Jar file. This means that several applications located in the same directory can share the same external files.

The application files are distinguished by unique names but share the common app directory. This can be a powerful tool if used for data/configuration sharing between applications.

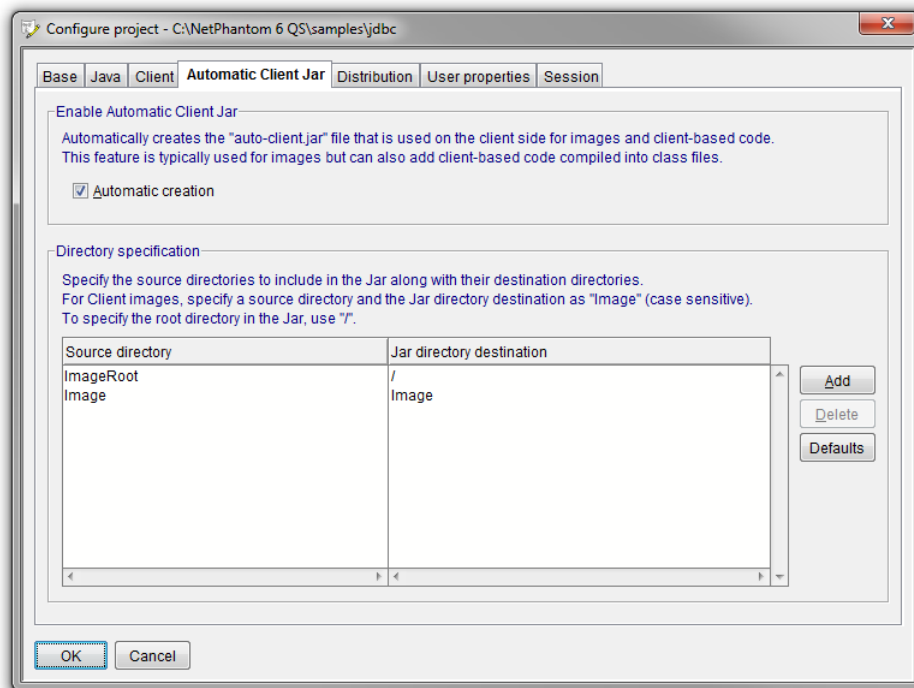
3.8 Automatic Client Jar

The Automatic Client Jar feature creates folders in the project that is used for the NetPhantom Client, typically images, but also Client-based code such as User Windows (the NetPhantom Swing and Java2D samples uses this feature).

To add a resource to the Client Jar, just drop it in the defined folder, e.g *Image*. The NetPhantom Builder then creates the Client Jar automatically, and signs it (signing is an option that is enabled by default). This is done transparently in the background, and when the NetPhantom Client is started, this Jar file (called `auto-client.jar`) is ready and loaded on the Client side.

Configuring Automatic Client Jar

Enable the feature from the *Configure project* dialog box on the *Automatic Client Jar* page:



Define the **Directory specification** as one or more directories in the project that should be placed in the Jar file, along with the destination directory inside the Jar. To place files in the root of the Jar, specify a forward slash '/' as shown above for *ImageRoot*.

4 Using NetPhantom and Eclipse

This chapter describes the various functions that are specific to the NetPhantom and Eclipse environments. For Eclipse specific functions, see its documentation, otherwise the *NetPhantom Administrator's and Developer's Guide*.

4.1 NetPhantom and Eclipse coexistence

NetPhantom and Eclipse are required to communicate in real time to coexist and integrate properly. They are, however, independent processes, so they may run stand-alone. The Eclipse workspace is the sharing point, and Eclipse locks it whenever in use.

NetPhantom Editor can use a project that is shared with Eclipse by having the reference to the Eclipse workspace location configured. This causes NetPhantom Editor to request a lock of the workspace before beginning an operation, such as a Build operation. If the lock can be obtained, i.e. if Eclipse is not running or is not using that workspace, NetPhantom will compile REXX and Java code. When the lock cannot be obtained, the console will show a message about it and let the Eclipse Builder run instead (the NetPhantom plug-in registers a REXX Builder and a post-Java Builder to handle compilation).

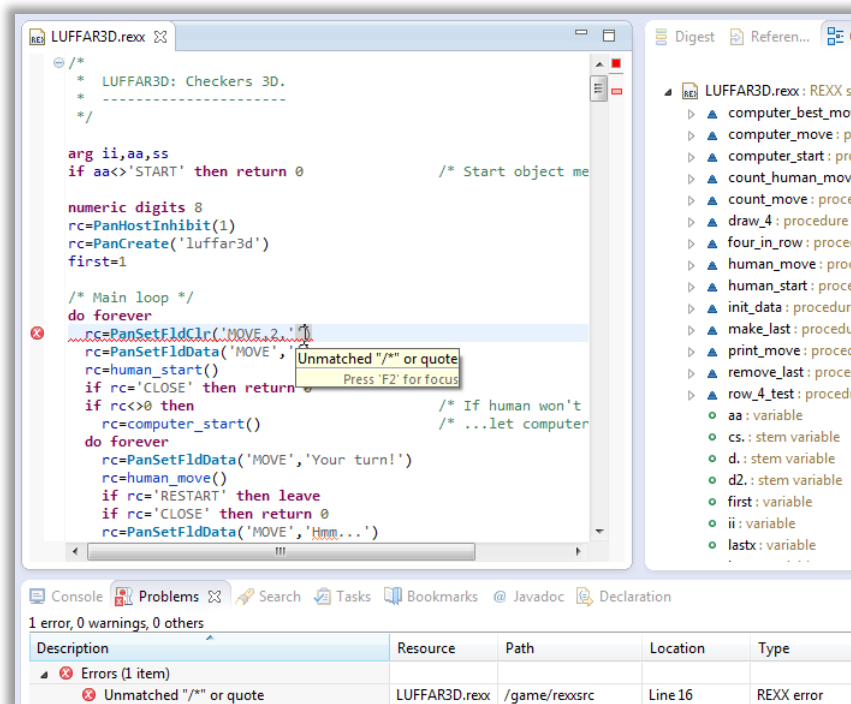
Several functions are added to facilitate the integration, i.e. to enable NetPhantom to launch Eclipse, verification that the NetPhantom plug-ins are up-to-date. Eclipse supports launching NetPhantom Editor at start and when the NetPhantom toolbar icon for connection is pressed.

The Eclipse icon in the menu bars of the Editor Terminal and Panel parts is enabled showing that the connection to Eclipse is established. If the icon is grayed, this indicates no connection, and pressing the icon will launch Eclipse if the process is not already running.

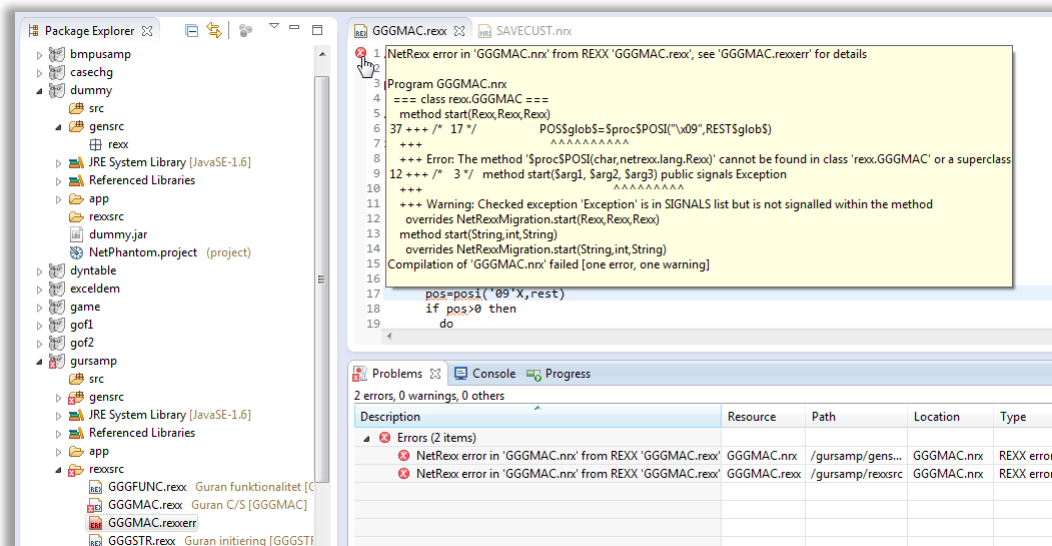
The icons for the Terminal part, the Panel part and Eclipse are otherwise used to activate its respective window as the desktop does not have room for all of them at once.

4.2 REXX Compilation in Eclipse

When REXX is compiled in Eclipse, when a REXX file is saved (and **Build automatically** option is enabled), all errors that can be directly related to the REXX source is marked with errors directly in the source editor of the erroneous line. These are typically errors such as syntax errors.



As REXX is compiled into NetRexx that has a different structure, the error cannot automatically be reported to an accurate location in the original REXX source. A `.rexterr` file is therefore created (with a red file icon) containing the textual error message in full. This file can be opened by simply double-clicking on the file. The Eclipse **Problems** view is also populated with compilation problems.



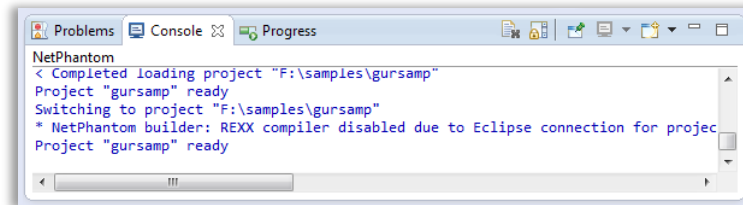
Example:

An error in `GGGMAC.rexx` calling the function `pos1(...)` instead of `pos(...)` would result in the above scenario. In the Explorer, the `GGGMAC.rexterr` file is created containing the text that is shown in the yellow tooltip that is displayed when the mouse pointer is moved of the error icon of line 1 in the source code. The Problem view gets two entries for the error, one that is placed in the `GGGMAC.rexx` source and one in the generated NetRexx source `GGGMAC.nrx` in the `gensrc/rexx` directory.

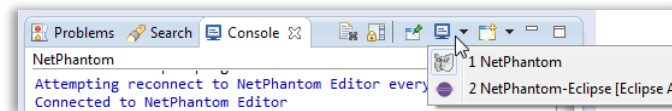
4.3 NetPhantom Console

NetPhantom generates many events from its different parts, such as the Server, loading an Application, launching a Client. This becomes even more intensive when Eclipse is given the control over project switching among other.

The NetPhantom Java console is therefore extended into the Eclipse NetPhantom Console, shown in the standard place of consoles in Eclipse.



When e.g. a Client is launched, a second console is allocated by Eclipse and both consoles utilize the same location for its contents. Switching between the console contents is done using the Console view toolbar button shown beside the mouse pointer:



The format of the logs can be configured by opening the **Console output properties** dialog in the **Java Console** window. The dialog opens by selecting **Edit – Properties**.

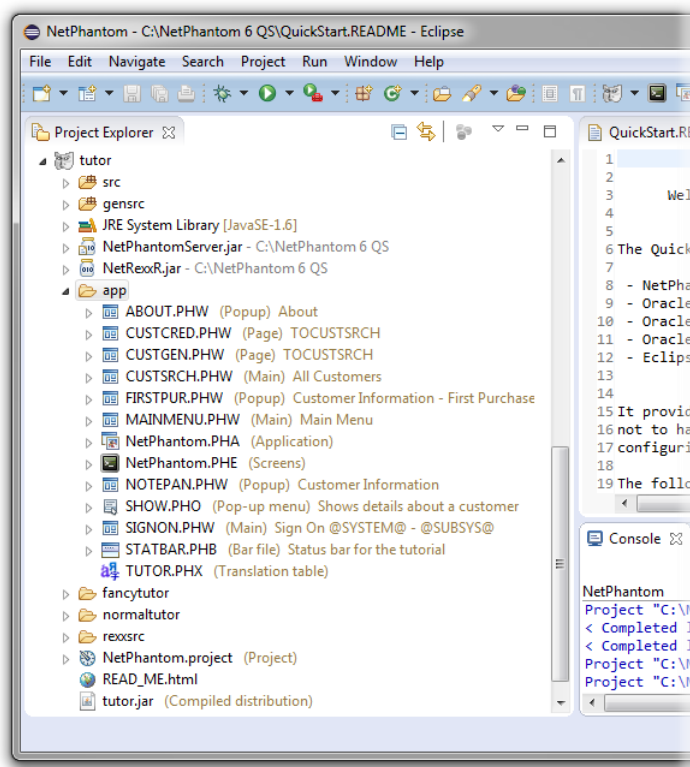
Optionally, the displayed log can be saved to file, specified by the **Log file name**. The visual representation of the log displayed in the windows can be modified in several ways. The **Line count** gives the maximum number of lines displayed before it is truncated, and the **Line width** specifies the maximum width (in characters).

The colors used to represent different types of log messages can be selected, per type. The font of the text in the log can be selected: **Font name** and **Size**.

All changes in the configuration of the visual representation of the log are immediately and retroactively visible both in the NetPhantom Editor and in Eclipse.

4.4 File structure

When defining a NetPhantom project together with Eclipse, it means that both tools work on the same structures and definitions. The following file structure is used in the Eclipse Workbench.



All files in the Eclipse **Project Explorer** (or the **Package Explorer**) are followed by descriptions them that are retrieved from the Editor, making it easier to locate an item. This is combined with the **Filter** function (used above to filter on the text ST entered in the Filter text field at the top middle of the screen capture above). The *Filter function* is described in detail below.

Note that the *Package Explorer* does not display detailed structure for NetPhantom, only *Project Explorer*.














Description of all the entities in the file structure

Bold items are required for the structure to be viable:

NetPhantom.project	NetPhantom project definition file.
.project	Eclipse project definition file.
.classpath	Eclipse Java class path definition.
app	Directory containing NetPhantom specific entities.
NetPhantom.PHA	Application definitions.
NetPhantom.PHE	Host definitions.
NetPhantom.PHH	Help texts definition.
filename.PHW	Panel definition.
filename.PHX	Translation table.
filename.PHM	Message table.
filename.PHB	Tool- and status bar definitions.
filename.CMB	Combobox definitions.
rexxsrc	REXX source files .rexx.
src	Java source files .java.
bin	Output directory for generated items.
gensrc	Intermediate directory for generated source items (used for debug).


4.5 NetPhantom Editors in Eclipse

The following file extensions are registered as internal or external editors by the NetPhantom plug-in. This enables quick access to the NetPhantom Editor file in question, or a direct open of the corresponding Eclipse editor. The table below shows the file icon associated with the file extension and what happens when you open the file (i.e. double-click on it) in the Explorer.

Icon	Type	Ext	Description
	Project	project	Opens the Project settings dialog.
	Application	PHA	Opens the Application data dialog box.
	Terminal	PHE	Activates the Terminal part.
	Bar file	PHB	Opens the tool- or statusbar dialog box.
	Help ID file	PHH	Opens the help ID file in a text editor.
	Text table	PHM	Opens the text table file in a text editor.
	Pop-up menu	PHO	Opens the pop-up menu definition.
	Translation	PHX	Opens the translation table file in a text editor.
	Combobox	CMB	Open the combobox file in a text editor.
	REXX source	rexx	Opens the REXX source in a text editor.
	REXX error file	rexxerr	Opens the REXX error file in a text editor.
	NetRexx source	nrx	Opens the NetRexx source file in a text editor.
	Java source	java	Opens the Java source in a Java editor.

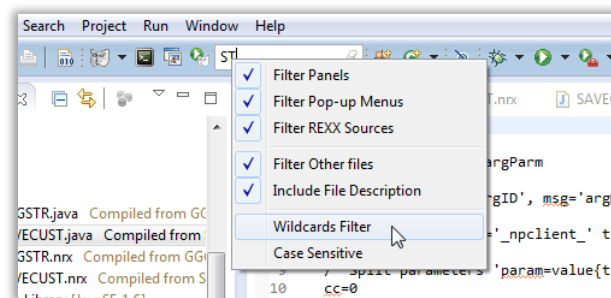
4.6 Open Object Source from Editor

The source code for an Object, i.e. the REXX code or the Java code, can be opened directly in Eclipse from NetPhantom Editor. There are several shortcuts available. When the **Object** entry field for a *Control*, *Menu item* or *Panel* is displayed in a dialog box, right-click to display the context menu. From there, select the menu item **Edit object source**.

The Eclipse source code can also be opened in the **Object definition** dialog box with .

4.7 Filter function

An Eclipse filter is added with the NetPhantom plug-in. It is used in the **Package Explorer** and the **Project Explorer** and is enabled by default.



The entry field in the NetPhantom toolbar is used to enter a short text string that is used for filtering items in the Explorer tree view. What items that participate in the filtering are selected from the pull-down menu. This includes **Panels**, **Pop-up Menus**, **REXX Sources** and **Other files** (all other files, i.e. not panels, pop-up menus or REXX sources).

The option **Include File Description** also uses the descriptive text after the file in the text filtering.

By default, the filter text is the specified string, with the option of **Case Sensitive**, meaning that the text is located somewhere in the text (and perhaps the description) of the item in question. This is a quick and easy way of finding a file.

Note: a file item that does not have a descriptive text is most certainly a file that is not directly part of the project distribution.

A more advanced filtering can be enabled with **Wildcards Filter**, i.e. that the string specified consists of the Windows Wildcards * ? ^, where * is one or more character, ? is one character and ^ escapes the next character (e.g. if you wish to search for * you enter ^*).

Other NetPhantom filters

There are other filters in the NetPhantom plug-in:

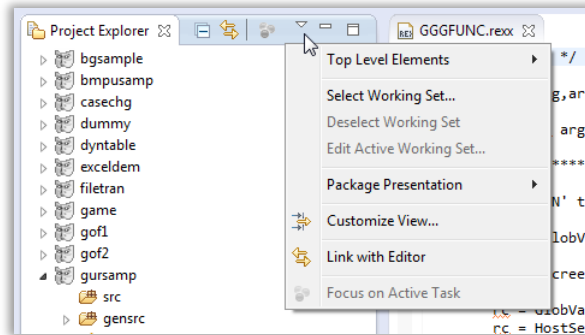
- Non-NetPhantom projects, hide non-NetPhantom projects.
- NetPhantom generated files, hides generated source directories files (`gensrc`).
- NetPhantom backup (GUI) files, hides backup files (`*._H*`), active by default.

Configuration

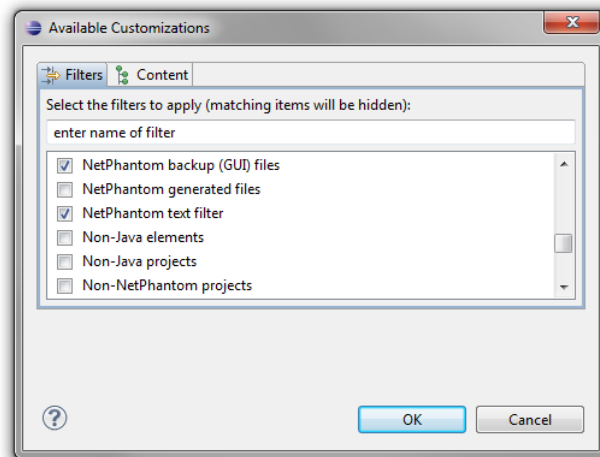
Configuration of the filters depends on the Explorer view, see below. Each filter type can be activated separately.

Project Explorer

The Project Explorer uses another dialog box, accessed in the same way, by selecting the down-arrow to show the drop-down menu and then selecting **Customize View**.



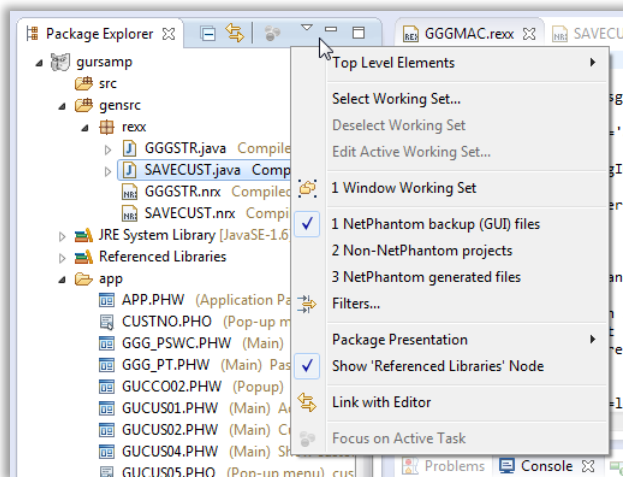
The NetPhantom filters are shown in the default activation states. Please note the filter **NetPhantom text filter** that must be activated for text filtering to work with the NetPhantom toolbar filter entry field.



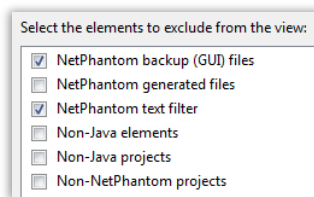
Package Explorer

This view is mainly used for Java as it doesn't show the deep structure of NetPhantom elements as Project Explorer does.

Package Explorer uses the Java Element Filters dialog box for configuration, shown by selecting the down-arrow to show the drop-down menu and then selecting **Filters**.

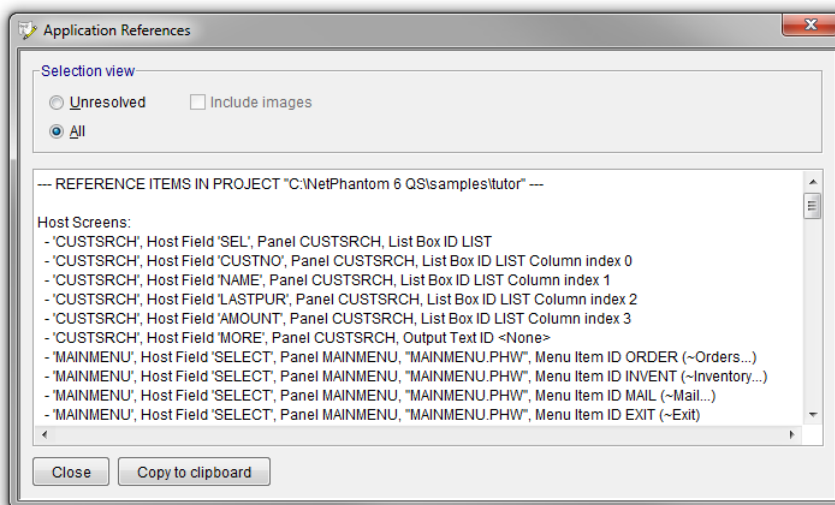


The list box in the dialog is shown below where the NetPhantom plug-in filters can be activated or deactivated. The states below are the default states.



4.8 Application references

The references in the project to screens, fields, panels, controls and menu items, pop-up menus, objects, text IDs, combination box files, bar files and images can be viewed using NetPhantom Editor terminal part menu item **File – Application references**.

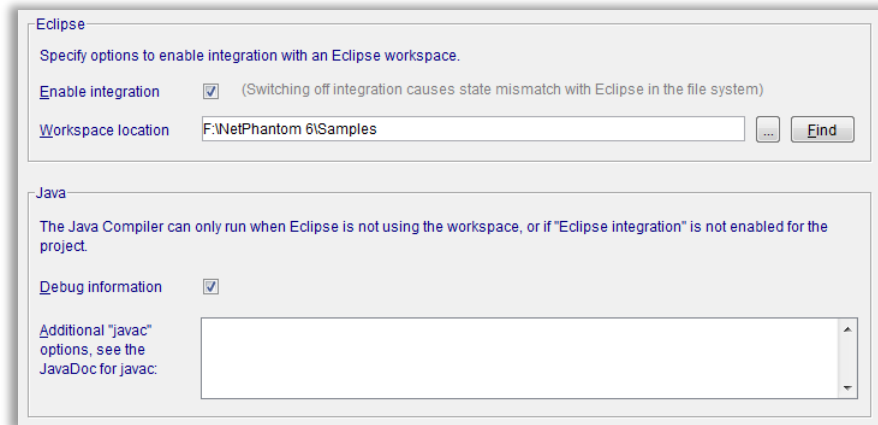


The dialog box can be used to view unresolved items as well as the full reference list, optionally to include unresolved images.

4.9 Eclipse and Java settings for NetPhantom projects

The NetPhantom project has the configuration for Eclipse integration and the Eclipse workspace configured in the project configuration. To display the dialog box, select menu item **File – Configure project**.

The **Java** page is used for the Eclipse settings as well as the Java Compiler, used when compiling a distribution.



4.10 Eclipse conversion of Java Project to NetPhantom

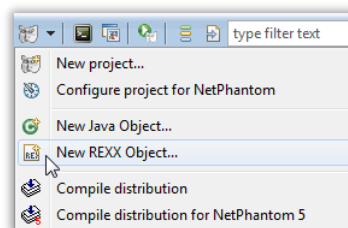
An existing Java project in Eclipse can be converted to a NetPhantom project. In this process NetPhantom specific files/directories are added to the existing Java project.

To initiate the conversion, right-click in the Package or Project Explorer and select the project in question, then the menu item **Configure – Configure Java project for NetPhantom** or select the same drop-down menu item from the NetPhantom toolbar.

Note that this does not take away any functionality from the Java project. From the Eclipse point of view, a NetPhantom 6 project is a superset of the corresponding Java project.

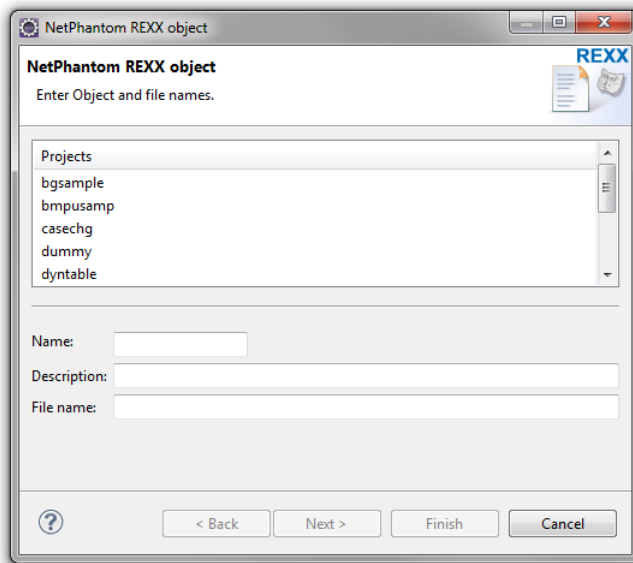
4.11 Creating a NetPhantom Object in Eclipse

New NetPhantom Objects can be created in Eclipse from the NetPhantom toolbar drop-down menu **NetPhantom REXX Object** or **NetPhantom Java Object**.



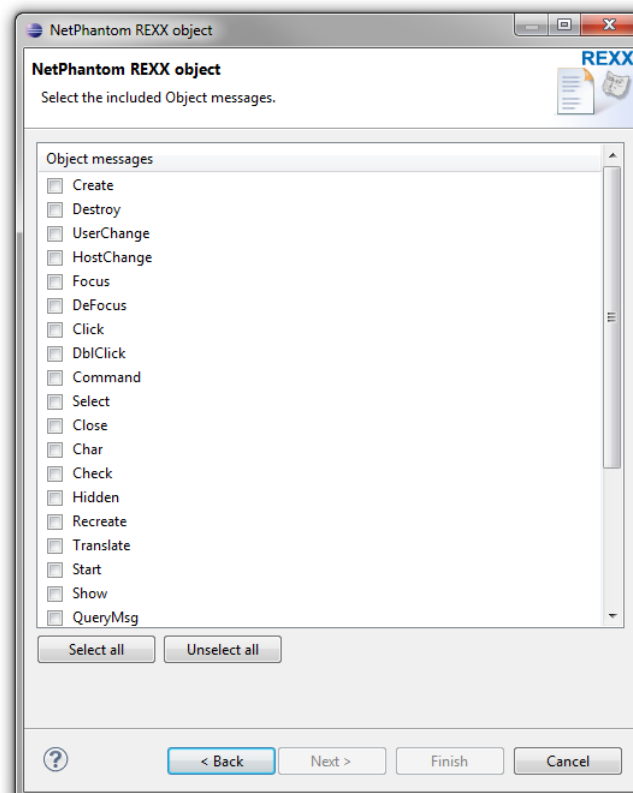
NetPhantom REXX Object

When defining a REXX Object, the following dialog is shown:

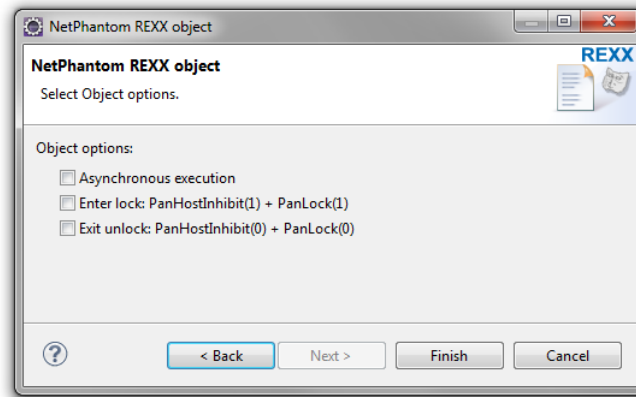


Select the project, enter the **Name**, **Description** and **File name** of the REXX source file. The file will be placed in the `rexsrc` directory and should have the file extension `.rexx`.

In the second page in the wizard, select one or more of the **Object messages** that should trigger the REXX code execution.



The last page in the wizard is used to define the execution Object options. The option **Asynchronous execution** causes the REXX code to execute asynchronously from the main thread (called REXX Application prior to NetPhantom 6, whereas REXX Macro is when the option is unselected).



The options for lock cause the GUI to be locked and inhibited from host processing upon entering the execution, and/or released upon exit. This simplifies coding so that the functions `PanHostInhibit` and `PanLock` are guaranteed to be called before and/or after the REXX code, no matter if executing asynchronously. This guarantee could be obtained by having two REXX Objects prior to NetPhantom 5, one being a REXX Macro that called `PanHostInhibit(1)` and `PanLock(1)` then calling a REXX Application to perform the execution when e.g. waiting for host response is required.

Once the **Finish** button is pressed the REXX Object is created and the REXX source is shown in the Eclipse text editor. The REXX template is used to create the REXX source stub code.

NetPhantom Java Object

The Java Object has been enhanced in NetPhantom 6 to provide a panel and/or component listener in addition or instead of the previous Object Message style calling interface.

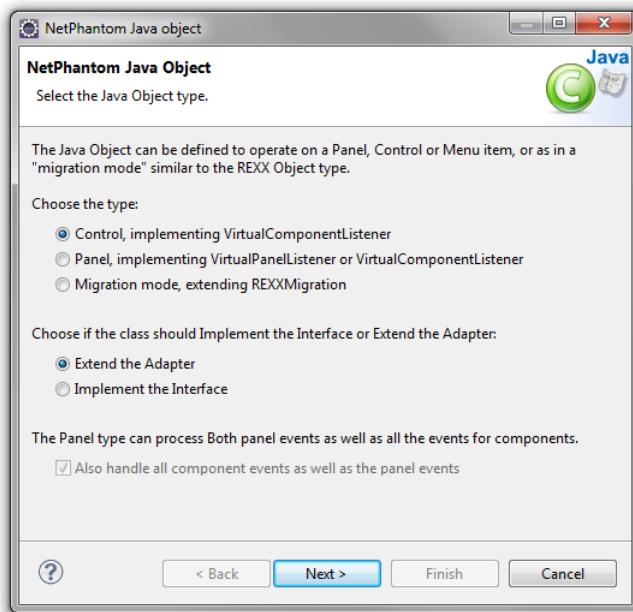
The new interfaces create an instance of the Java class that lives if the panel it is connected to is present. A Java class of this type should be attached to a *Panel* or the *Notebook Page Panel*, and *NOT* to *Controls* or *Menu items*. The implementation of the Java code can be done using a Java *interface* or by *extending a class*.

Two flavors of the interface are available:

- The interface `se.entra.phantom.server.VirtualPanelListener`, or
- by extending the class `se.entra.phantom.server.VirtualPanelAdapter`.

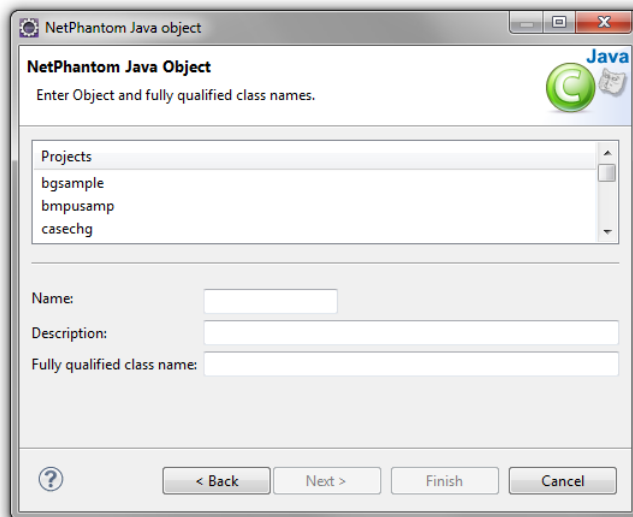
It is recommended to use the extend mechanism of the adapter rather than implementing the interface, the adapter could be enhanced in future versions of NetPhantom, whereas the interface would require to have e.g. a second interface to be implemented by the class as well.

Creating a new Java Object opens the NetPhantom Java Object wizard:



In this dialog, select the Java Object implementation type: it is recommended to **Extend the Adapter**, and the **VirtualComponentListener** provides the largest set of events from the panel, its controls, or its menu items.

In the second page of the wizard, select the project and specify the Object **Name**, **Description** and **Fully qualified class name**.



Once the wizard has completed, the Java source code stub is created from templates with the selected implementation type and opened in the Eclipse Java source code editor.




5 Application Execution

The application under development can be launched for testing purposes. This will create a temporary in-memory distribution to run on the internal NetPhantom Server. Even applications containing errors can be launched. Answer **Yes** in the warning message box and the application will be launched in a “*best effort*” mode.


There are three ways to run the application:

1. Java application.
2. Java application in debug mode.
3. Applet launched in a browser.

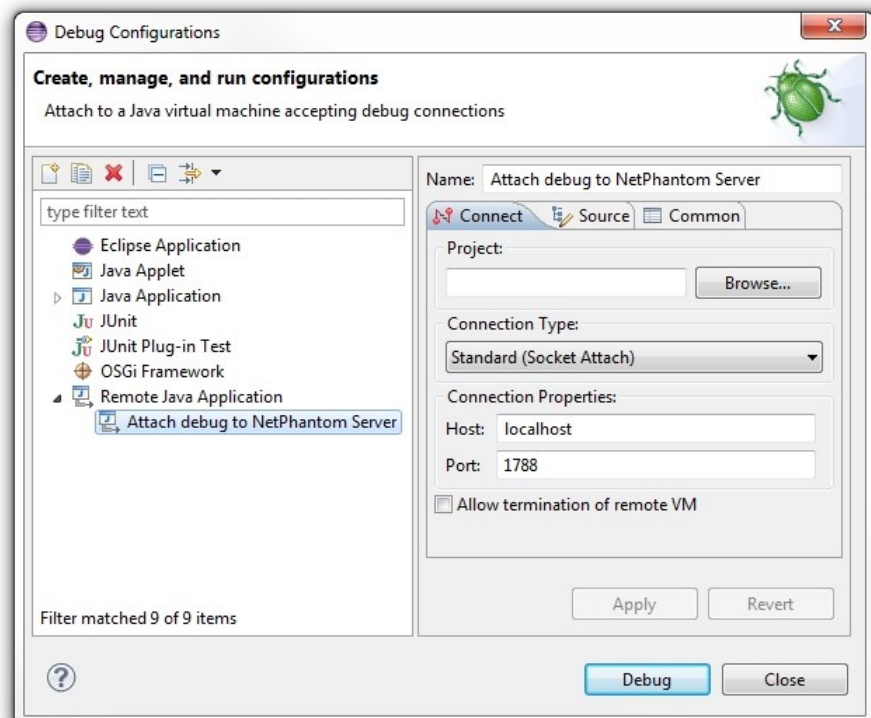
There are many ways to launch the application. The Client can be launched from all windows:

-  Terminal part,
-  Panel part,
-  Eclipse, on the NetPhantom toolbar.

When running the Client from Eclipse, *project switching is enabled*. A new feature in NetPhantom 6 is also that there is *no limit as to how many simultaneous client sessions started* (NetPhantom 5.5 only allowed one client).

To run the application in debug mode using the NetPhantom Terminal part icon  is still available for compatibility. This launches the NetPhantom Client for remote debugging of the Client itself and all code running there. This is mostly used when developing Client User Windows, something that is not commonly used.

5.1 Debugging the Server application



To run the Server application in debug mode means that the Java code written can be debugged in the Eclipse debugger. Also, the REXX code can be debugged since the intermediate Java code is saved in the `gensrc/rexx` directory. In the Java code in that directory, breakpoints can be set.

Warning!

When setting breakpoints in the code great care should be taken. Any breakpoint will halt the thread it is running in; do not forget to release it back to running state. That means that the functionality of Server could be impaired.

As a convenience function, the NetPhantom Eclipse plugin will generate a default debug configuration for connecting to a running NetPhantom application. This launch configuration is called **Attach debug to NetPhantom Server**. To view this configuration, select the menu item **Run – Debug Configurations**.

The configuration to use is the one defined in the NetPhantom Editor as described in chapter **Error! Reference source not found.**. The configuration is transferred when the NetPhantom Editor and Eclipse connect. This means that changes might require restart to take effect.

Again: be careful with breakpoints as this may impair the Editor for a halted thread!

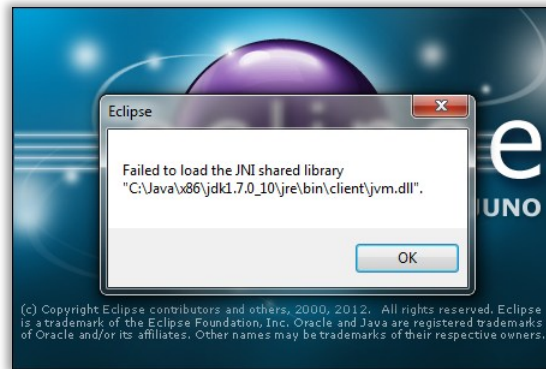
After pressing **Debug**, nothing much happens until you launch a NetPhantom Client that eventually hit a breakpoint you have set...

6 Troubleshooting

This section covers common troubleshooting scenarios.

6.1 Problems using 64-bit Eclipse with NetPhantom

If you use a 64-bit version of Eclipse, a problem to launch Eclipse from NetPhantom may exist in certain operating systems as the screen capture below shows. The Eclipse stub executable is launched, but then displays the error message on top of its splash screen.



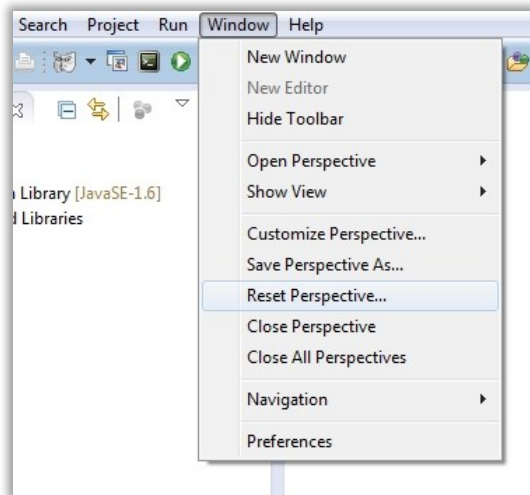
If this happens, open the `eclipse.ini` where `eclipse.exe` is located and enter the path to the `javaw.exe` executable, prefixed with a line `-vm` that defines the Java Virtual Machine executable:

```
-vm
c:\program files\java\jdk1.8.0_263\jre\bin\javaw.exe
-startup
plugins/org.eclipse.equinox.launcher_1.3.0.v20120522-1813.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.win32.win32.x86_64_1.1.....
-showsplash
org.eclipse.platform
--launcher.XXMaxPermSize
256m
--launcher.defaultAction
openFile
-product
org.eclipse.epp.package.jee.product
-vmargs
-Xms40m
-Xmx800m
```

6.2 NetPhantom menus, items or views do not appear

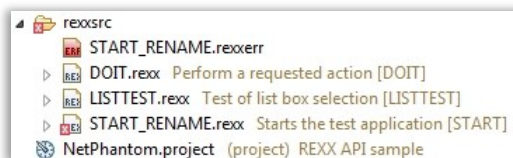
The NetPhantom plug-in is installed, but the workspace is missing many of the menus, items, toolbar and/or views.

1. Make sure to have the NetPhantom perspective open,
2. **Reset perspective:**

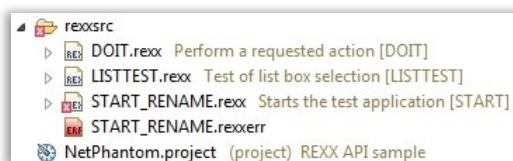


6.3 The NetPhantom Project layout is wrong

Sometimes project states are not maintained properly by Eclipse as in the example below.



The `START_RENAME.rexxerr` file contains the errors and warnings of the `START_RENAME.rexx` file. In a project with many source files, this way of sorting the files can be difficult to handle. Doing a refresh (F5 shortcut) on the project will give the following result.



The error files are located after their respective source file for convenience.

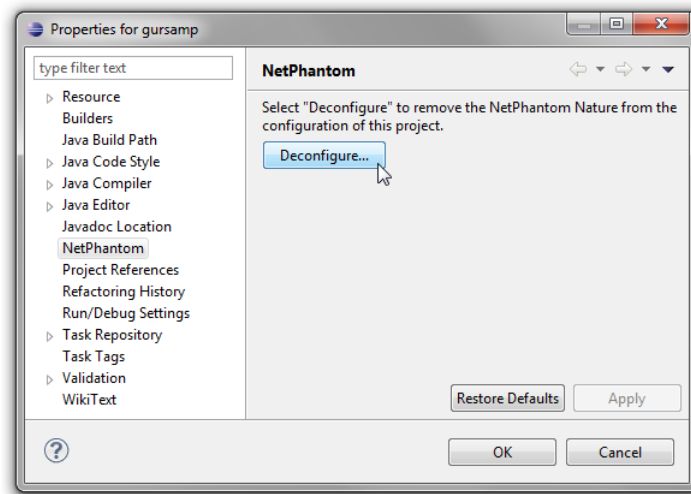
This specific example illustrates a more generic mechanism; when a project looks inconsistent in Eclipse, the first thing to try is to refresh the state.

6.4 The NetPhantom project is inconsistent

To reset the state in a project, two things can be done:

1. **Close and reopen** the project in question, combined with **Clean** operation.
2. **Deconfigure and reconfigure** the project as a NetPhantom project.

These solutions should be tried in that order. The way to deconfigure a project is done by selecting the project in Explorer, then right-click and, in the context menu, select **Preferences**. In the left tree, select **NetPhantom** and press **Deconfigure**.



The resulting project is of Java type and can be continued to be used as such. Please note that the NetPhantom files are not removed from the project. This means that no data is lost when deconfiguring a NetPhantom project.

The way to (re-)configure a Java project is described in Chapter 4.10 *Eclipse conversion of Java Project to NetPhantom*.

6.5 Connection between Eclipse and NetPhantom is disrupted

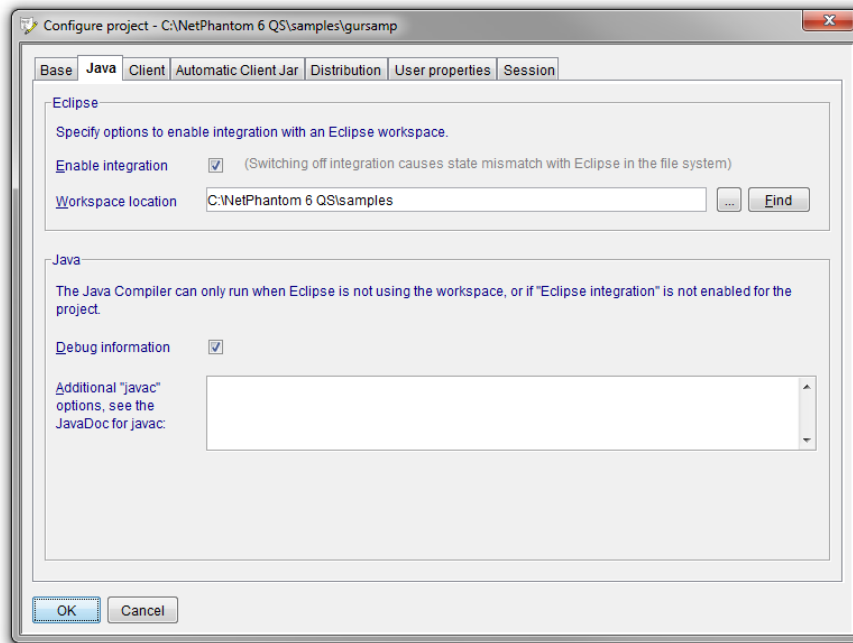
In the event of a total loss of communication or as a subset of integrated function stops working:

1. Check if there are errors or warning messages in the NetPhantom Console.
2. Check if there are modal dialogs open (even under other open windows). These could block functions in a non-transparent way. Closing the dialog should unlock the communication.
3. Disconnecting and reconnect from Eclipse should reinitialize the communication. If this fails, the most reliable method is to close the NetPhantom Editor and then open it again by connecting from the Eclipse side. This is done by clicking the NetPhantom menu in the toolbar and answering affirmative to the message box asking if the NetPhantom Editor should be re-started.

6.6 The Java code is compiled with errors or warnings

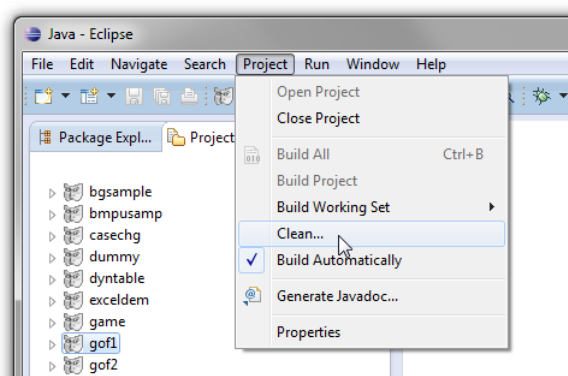
This occurs even if there should not be any errors or warnings and could be due to conflicting parallel compilations (Eclipse/NetPhantom).

Enable **Eclipse integration**:



Without selecting the **Eclipse integration** option, the two builders/compiler will compete for the access of the source code and the access to write in the output directory. Depending on the compile configurations, this will, in most cases cause inconsistency.

After making the change above, the application should be cleaned in Eclipse using the menu item **Project – Clean**:



Note: This cleaning must now be done through Eclipse. This is because Eclipse is the party responsible for compilation at this point. An attempt to clean the project through NetPhantom Editor will result in an error message.

6.7 Load error “Unsupported ClassVersionError” in application

When launching a Client to run the application, the Server (inside the Editor) loads the application and shows an error like:


```
java.lang.UnsupportedClassVersionError: classname:
    Unsupported major.minor version 51.0
```

This error is logged when an older Java VM loads a class that is compiled for higher Java version.

In general, this would occur when the Eclipse workspace or project uses Java 14 thus Eclipse compiles the code that cannot be loaded by a Java 1.8 engine.

The solutions to the problem are:

1. Verify that Eclipse and NetPhantom are configured to coexist, i.e. that the project is configured for Eclipse workspace sharing and integration.
2. Check the Java versions in the Editor and the Eclipse workspace *and* project: make sure Java 1.8 is configured.

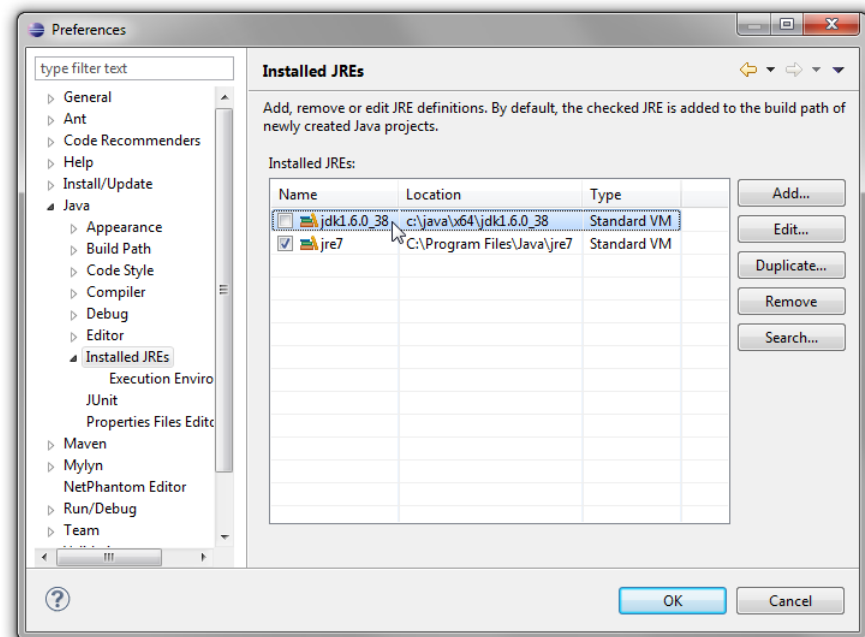
6.8 The NetPhantom Project is marked with a warning

The warning message is:

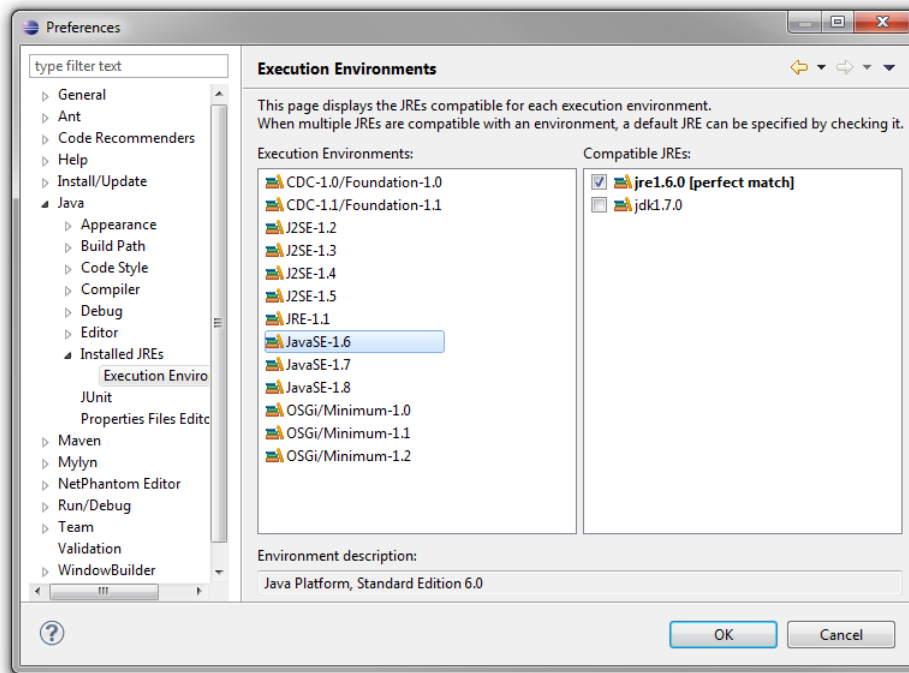
Build path specifies execution environment JavaSE-1.6. There are no JREs installed in the workspace that are strictly compatible with this environment.

You are using Eclipse with a JRE of version 1.8 or better. Define an installed JRE according to recommendations.

Add Java Runtime Environment for Java 1.8, Java 11 and possible higher version:



Also make sure to define the **JavaSE-1.8** to the newly installed Java 1.8 JRE/JDK on the **Execution Environment** page (same goes for Java 11 and up):



Note: Eclipse doesn't show you the newly installed JRE/JDK in the list of **Compatible JREs** when you have just installed the JRE/JDK. You must first close the dialog box and reopen it. The JRE/JDK just installed will then appear.

6.9 Warning "bootstrap class path not set..."

When the NetPhantom builder is active or when you compile a distribution the warning message "<JavaCompiler>: bootstrap class path not set in conjunction with -source 1.6".

```
* NetPhantom builder: REXX compiler enabled, Eclipse is not integrated with project
> Build: SRVAUTO.rexx STARTUP.rexx SENDMAIL.rexx
< Build: 3 sources successfully compiled
> Java compiler: gensrc\rexx\STARTUP.java gensrc\rexx\SRVAUTO.java gensrc\rexx\SENDMAIL.java
<JavaCompiler>: bootstrap class path not set in conjunction with -source 1.6
```

The Editor sometimes uses a JDK version 1.6 and sets the compiler options `-source 1.6` and `-target 1.6` automatically because the distribution must be able to run on a NetPhantom Server running Java 1.6. To get pure 1.8 Java compliance, you must edit the classpath of the Java environment and set the library Jars to the ones from a JDK 1.8. This is quite complicated and errorprone, thus we recommend configuring the project to use Java 8 instead.

6.10 Error "Panel MODAL disposed..."

Running an application which opens a main panel from a modal pop-up panel will bring the application down.

```
java.lang.InternalError: $$$ ERROR: Panel MODAL disposed while in modal process!
at se.entra.phantom.server.VirtualPanel.dispose(Unknown Source)
at se.entra.phantom.server.VirtualPanelSession.setPanelNestLevel(Unknown Source)
at se.entra.phantom.server.VirtualPanelSession.a(Unknown Source)
```

The reason for this is that logically, the main panel should remove any other open panel. This cannot be met since the pop-up panel is modal and as such has locked the hierarchy.

To fix this problem, simply change the opened panel type from **Main** to **Pop-up**.

6.11 Error “The system is out of resources”

As described in this document, due to the available functionality, the integrated development requires a big number of resources in terms of memory. When memory runs out, the system will log or prompt a message as below:

```
The system is out of resources.
Consult the following stack trace for details.
java.lang.OutOfMemoryError: Java heap space
```

Due to the complexity of the environment, this can happen in various places in the system.

The first thing to consider in this situation is to close projects in the Eclipse workspace. All open projects will require allocated memory in both Eclipse and the NetPhantom Editor. Closing the project will reclaim that memory.

If that is not possible or practical, there are several places where memory limits can be increased.

Eclipse

The memory configuration of Eclipse is in the `eclipse.ini` file in the installation directory (for Quick Start, `C:\NetPhantom 7 QS\eclipse` by default). The structure of this file is a bit particular:

1. Each option and each argument to an option must be on its own line.
2. All lines after `-vmargs` are passed as arguments to the JVM, so all arguments and options for eclipse must be specified before `-vmargs` (just like when you use arguments on the command-line)

To increase the initial amount (*ms*) and the upper limit (*mx*) allocation of the Java virtual machine running eclipse, the following should be added to the end of the file:

```
-vmargs
-Xms256m
-Xmx900m
```

As Eclipse with NetPhantom 7 (or better) is a 64-bit process, the memory limit is much higher and may be set to Giga Bytes (**g**), for example:

```
-vmargs
-Xms400m
-Xmx2g
```

NetPhantom Editor

The configuration of The NetPhantom Editor is in the `PHANTOM.INI` file in the installation directory (for Quick Start, `C:\NetPhantom 7 QS` by default). The section handling the Java virtual machine parameters is called Java:

```
[Java]
option.1=-Djava.class.path=NetPhantomServer.jar;...
option.2=-Xmx900m
option.3=-Xms128m
```

Options 2 and 3 are the ones used to configure the memory allocation. This setting is almost set to a maximum for a 32-bit JVM, i.e. 900 MB, so there is not much room to increase it as 1050 MB is the practical limit.

NetPhantom Application

The configuration of an application launched for test purposes is also located in the `PHANTOM.INI`. However, these parameters can be accessed through the Administration interface by selecting **Options – Configure Java environment**. In this dialog the `-Xms` and `-Xmx` parameters can be configured as described above.